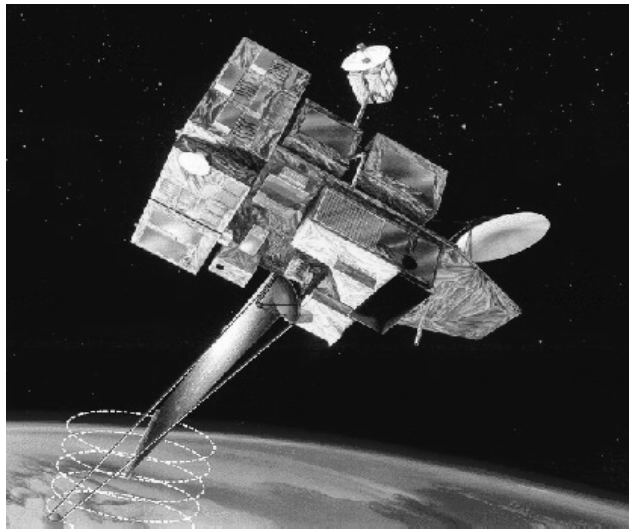


686-631

Science Algorithm Specification
for
SeaWinds on QuikSCAT
and
SeaWinds on ADEOS-II



R. Scott Dunbar
S. Vincent Hsiao
Young-Joon Kim
Kyung S. Pak
Barry H. Weiss
Angela Zhang

Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, California 91109

JPL D-21978

October 5, 2001

Science Algorithm Specification
For
SeaWinds on QuikSCAT
and
SeaWinds on ADEOS-II
686-631

Science Algorithm Manager: Philip S. Callahan

Document Editor: R. Scott Dunbar

Science Algorithm Authors: R. Scott Dunbar
Vincent Hsiao
Young-Joon Kim
Kyung S. Pak
Barry H. Weiss
Angela Zhang

JPL D-21978

JPL

Jet Propulsion Laboratory
California Institute of Technology

Table of Contents

| | |
|--|-----------|
| Preface | v |
| SeaWinds/QuikSCAT Algorithm Heritage Table..... | vi |
| SeaWinds Spacecraft Location and Nadir Location Algorithm | 2 |
| L1A.1.1.1/L1B.1.1.1 Initialize_Ephemeris | 4 |
| L1A.1.1.2/L1B.1.1.2 Read_Ephemeris | 7 |
| L1A.1.1.3/L1B.1.1.3 Spline | 10 |
| L1A.1.2.1/L1B.1.2.1 Interpolate_Ephemeris | 12 |
| L1B.1.2.2 Spline Interpolator..... | 15 |
| L1B.1.3.1 Nadir..... | 17 |
| SeaWinds Geometry Algorithms | 22 |
| L1B.2.2.1 Compute_Local_Coord | 28 |
| L1B.2.2.2 Compute_Attitude_Rotatn_Matrix | 30 |
| L1B.2.2.3 Convert_SC_To_Local..... | 32 |
| L1B.2.2.4 Convert_Between_Rect_Local..... | 34 |
| L1B.2.2.5 Convert_Rect_To_Geo..... | 36 |
| L1B.2.2.6 Compute_Geodetic_Lat..... | 37 |
| L1B.2.3.1 Compute_Max_Gain_Dir..... | 38 |
| L1B.2.3.3 Determine_Measurement_Time..... | 39 |
| L1B.2.4.1 Locate_Cell_On_Earth..... | 42 |
| L1B.2.4.2 Compute_Incidence_Angle | 44 |
| L1B.2.4.4 Compute_Cell_Azimuth_Angle..... | 46 |
| L1B.2.4.5 Doppler_Shift_Range_Track..... | 48 |
| L1B.2.4.6 Compute_X_Loc | 52 |
| L1B.2.4.7 Select_Best_Eight_Slices..... | 56 |
| L1B.2.4.8 Locate_Freq_Lines | 58 |
| L1B.2.4.9 Locate_Peak_Gain..... | 61 |
| SeaWinds Sigma0 and Kp Algorithm..... | 64 |
| L1B.3.1.1 Process_Calibration_Data | 71 |
| L1B.3.2.1 Compute_Sigma0_and_Kp | 75 |
| L1B.3.3.1 Get_Cal_Data | 79 |
| L1B.3.3.2 Est_Calibration_X..... | 82 |
| L1B.3.4.1 Calculate_Pr_Pt_Ratio | 87 |
| L1B.3.4.2 Est_Noise_Energy | 90 |
| L1B.3.4.3 Est_Signal_Energy | 94 |
| L1B.3.4.4 Est_SNR..... | 96 |

| | |
|--|------------|
| L1B.3.4.5 Est_Pr_Pt_Ratio | 99 |
| L1B.3.5.1 Calculate_X_Factor..... | 102 |
| L1B.3.5.2 Calculate_Sigma0..... | 109 |
| L1B.3.5.3 Calculate_Kpc_Coeff | 113 |
| SeaWinds Scatterometer Brightness Temperature Algorithm | 120 |
| L1B.4.1.1 Calculate_Tb_Parameters..... | 123 |
| L1B.4.1.2 Calculate_Brightness_Temperature | 127 |
| Track Echo Signal | 132 |
| L1B.5.1.1 Track_Echo_Signal | 136 |
| L1B.5.2.0 Manage_Echo_Track_Frames..... | 141 |
| L1B.5.2.1 Initialize_Echo_Track | 144 |
| L1B.5.2.2 Track_Frame_Times | 145 |
| L1B.5.3.0 Acquire_Echo_Track_Matrix..... | 146 |
| L1B.5.3.1 Ascertain_Beam_Parameters..... | 150 |
| L1B.5.3.2 Determine_Echo_Track_Params..... | 152 |
| L1B.5.4.0 Calculate_Echo_Track_Attitude | 154 |
| L1B.5.4.1 Determine_Singularity | 158 |
| SeaWinds Scatterometer Calibration Smoothing Algorithm..... | 161 |
| L1B.6.1.1 Initialize_Cal_Pulse | 163 |
| L1B.6.1.2 Process_Calibration_Data | 170 |
| L1B.6.1.3 Get_Cal_Data | 180 |
| Level 2A /Level 2B ALGORITHMS | |
| Grouping Sigma0 and Calculating Sigma0 Variances..... | 184 |
| L2A.1.1.1 Regroup_Sigma0..... | 188 |
| L2A.1.2.1 Compute_Orbit_Element | 192 |
| L2A.1.3.1 Sub-track Binning | 195 |
| L2A.1.3.2 Bin_Sigma0_Parameters | 200 |
| L2A.1.4.1 Estimate_Sigma0_Variance | 203 |
| L2A.1.5.1 Collect_Brightness_Temps | 206 |
| L2A.2.1.1 Determine_Composites..... | 209 |
| L2A.2.1.2 Determine_Slice_Quality..... | 211 |
| L2A.2.1.3 Get_Composite_Usability | 213 |
| L2A.2.1.4 Composite_Slices..... | 215 |
| L2A.2.1.5 Composite_Sigma0_Parameters_Sig | 218 |
| L2A.2.2.1 Calculate_Tb_Statistics..... | 221 |

| | |
|--|------------|
| L2A.3.1.1 Flag_By_Mask | 223 |
| L2A.3.1.2 Flag_Read_Map | 225 |
| SeaWinds Sigma0 Grouping Algorithm..... | 228 |
| L2B.1.1.1 Execute_Wind_Algorithms | 232 |
| L2B.1.2.1 Prepare_WVC | 236 |
| L2B.1.2.2 Filter_WVC | 240 |
| L2B.1.2.3 Calculate_WVC_Centroid | 243 |
| L2B.1.2.4 Calculate_WVC_Brightness_Temp | 245 |
| L2B.1.2.5 Set_Wind_Flags | 247 |
| SeaWinds Wind Retrieval Algorithm..... | 252 |
| L2B.2.1.1 Retrieve_Winds | 256 |
| L2B.2.1.2 Calculate_Initial_WVC_Solution | 260 |
| L2B.2.1.3 Optimize_Wind_Solutions | 265 |
| L2B.2.1.4 Rank_Wind_Solutions | 273 |
| L2B.2.2.1 Evaluate_Objective_Function | 277 |
| L2B.2.2.2 Run_Model_Function..... | 281 |
| L2B.2.2.3 Initialize_Model_Function | 284 |
| L2B.2.2.4 Initialize_Interpolation_Coefficients..... | 286 |
| L2B.2.2.5 Initialize_Working_Table | 288 |
| L2B.2.2.6 Evaluate_Model_Function | 290 |
| SeaWinds Rain Flag Algorithm | 296 |
| L2B.3.1.1 Determine_Rain_Flag | 299 |
| L2B.3.1.2 Calculate_Normalized_Objective_Function_Rain_Index..... | 309 |
| L2B.3.1.3 Calculate_Transmittance | 315 |
| L2B.3.2.1 Filter_Rain_Flag_Spatially | 319 |
| SeaWinds Ambiguity Removal Algorithm..... | 323 |
| L2B.4.1.1 Execute_Ambiguity_Removal | 328 |
| L2B.4.1.2.1 Select_Ambiguity | 330 |
| L2B.4.1.2.2 Select_Nudge | 332 |
| L2B.4.1.2.3 Determine_NWP_Nudge | 334 |
| L2B.4.1.3 Initialize_Working_Arrays..... | 336 |
| L2B.4.1.4 Calculate_Median_Filter | 338 |
| L2B.4.1.5 Replace_Ambiguity_Buffers | 343 |
| Multi-Pass Ambiguity Removal for Reducing Rain Effects..... | 346 |
| L2B.4.1.1 Execute_Ambiguity_Removal (multi-pass) | 348 |
| L2B.4.1.3 Initialize_Working_Arrays (multi-pass) | 351 |

| | |
|--|------------|
| Direction-Interval Retrieval with Thresholded Nudging | 355 |
| L2B.5.1.1 Convert Objective Function to PDF..... | 359 |
| L2B.5.1.2 Build_Dir_Ranges | 363 |
| L2B.5.1.3 Correct_Selected_Vectors | 366 |
| L2B.5.1.4 DIR_Median_Filter | 369 |
| L2B.5.2.1 Thresholded Nudging | 372 |
| L2B.2.1.1 Retrieve_Winds (DIRTH) | 375 |
| L2B.2.1.2 Calculate_Initial_WVC_Solution (DIRTH)..... | 380 |

Preface

The *Science Algorithm Specifications for SeaWinds on QuikSCAT and SeaWinds on ADEOS-II* is a document describing the science data processing algorithms developed for the NASA's SeaWinds instrument from September 1997 through June 2001. Its contents are divided into four levels: Spacecraft geometry (L1A), cell geometry and sigma0 (L1B), regrouping sigma0 (L2A), and wind retrieval and ambiguity removal (L2B). Within each level there can be several algorithm modules. A module usually contains several lower-level algorithms that are related by the function performed. Each individual algorithm is called a subroutine and its description includes the formulation, processing steps, input/output data, and algorithm heritage.

The SeaWinds on QuikSCAT mission was launched on June 19, 1999 and has operated continuously to date. The SeaWinds on ADEOS II mission is due to launch in the early 2002.

SeaWinds/QuikSCAT Algorithm Heritage Table

N = New

MQ = Modified for QuikSCAT

MS = Modified for SeaWinds

| Algorithm ID | Algorithm Short (ATB) Name | QS launch | QS CalVal | SWS launch | Comments |
|-------------------------------|-----------------------------|--------------|--------------|---------------|---------------------------|
| Level 1A/1B Algorithms | | | | | |
| L1AB.1.1.1 | Initialize_Ephemeris | N | | | Used in L1A and L1B proc. |
| L1AB.1.1.2 | Read_Ephemeris | N | | | Used in L1A and L1B proc. |
| L1AB.1.1.3 | Spline | N | | | Used in L1A and L1B proc. |
| L1AB.1.2.1 | Interpolate_Ephemeris | N | | | Used in L1A and L1B proc. |
| L1AB.1.2.2 | SplInt | N | | | Used in L1A and L1B proc. |
| L1AB.1.3.1 | Nadir | N | | | Used in L1A and L1B proc. |
| Level 1B Algorithms | | | | | |
| L1B.2.2.1 | Compute_Local_Coord | N | | | |
| L1B.2.2.2 | Compute_Attitude_Rtn_Matrix | N | | | |
| L1B.2.2.3 | Convert_SC_To_Local | N | | | |
| L1B.2.2.4 | Convert_Between_Rect_Local | N | | | |
| L1B.2.2.5 | Convert_Rect_To_Geo | N | | | |
| L1B.2.2.6 | Compute_Geodetic_Lat | N | | | |
| L1B.2.3.1 | Compute_Max_Gain_Dir | N | | | |
| L1B.2.3.3 | Determine_Measurement_Time | N | | | |
| L1B.2.4.1 | Locate_Cell_On_Earth | N | | | |
| L1B.2.4.2 | Compute_Incidence_Angle | N | | | |
| L1B.2.4.4 | Compute_Cell_Azimuth_Angle | N | | | |
| L1B.2.4.5 | Doppler_Shift_Range_Track | N | | | |
| L1B.2.4.6 | Compute_X_Loc | N | | MS | |
| L1B.2.4.7 | Select_Best_Eight_Slices | | | N | |
| L1B.2.4.8 | Locate_Freq_Lines | N | | MS | |
| L1B.2.4.9 | Locate_Peak_Gain | N | | MS | |
| L1B.3.1.1 | Process_Calibration_Data | N | | | |
| L1B.3.2.1 | Compute_Sigma0_and_Kp | N | | | |
| L1B.3.3.1 | Get_Cal_Data | N | | | |
| L1B.3.3.2 | Est_Calibration_X | N | | | |
| L1B.3.4.1 | Calculate_Pr_Pt_Ratio | N | | | |
| L1B.3.4.2 | Est_Noise_Energy | N | | | |
| L1B.3.4.3 | Est_Signal_Energy | N | | | |
| L1B.3.4.4 | Est_SNR | N | | | |
| L1B.3.4.5 | Est_Pr_Pt_Ratio | N | | | |
| L1B.3.5.1 | Calculate_X_Factor | N | | | |
| L1B.3.5.2 | Calculate_Sigma0 | N | | | |
| L1B.3.5.3 | Calculate_Kpc_Coeff | N | | | |
| L1B.4.1.1 | Calculate_Tb_Parameters | | N | | |

| | | | | | |
|---------------------|---------------------------------|---|---|---|--|
| L1B.4.1.2 | Calculate_Brightness_Temp | | N | | |
| L1B.5.1.1 | Track_Echo_Signal | | | N | |
| L1B.5.2.0 | Manage_Echo_Track_Frames | | | N | |
| L1B.5.2.1 | Initialize_Echo_Track | | | N | |
| L1B.5.2.2 | Track_Frame_Times | | | N | |
| L1B.5.3.0 | Acquire_Echo_Track_Matrix | | | N | |
| L1B.5.3.1 | Ascertain_Beam_Parameters | | | N | |
| L1B.5.3.2 | Determine_Echo_Track_Params | | | N | |
| L1B.5.4.0 | Calculate_Echo_Track_Attitude | | | N | |
| L1B.5.4.1 | Determine_Singularity | | | N | |
| L1B.6.1.1 | Initialize_Cal_Pulses | N | | | |
| L1B.6.1.2 | Process_Calibration_Data | N | | | |
| L1B.6.1.3 | Get_Cal_Data | N | | | |
| Level 2A Algorithms | | | | | |
| L2A.1.1.1 | Regroup_Sigma0 | N | | | |
| L2A.1.2.1 | Compute_Orbit_Elements | N | | | |
| L2A.1.3.1 | Sub-track_Binning | N | | | |
| L2A.1.3.2 | Bin_Sigma0_Parameters | N | | | |
| L2A.1.4.1 | Estimate_Sigma0_Variance | N | | | |
| L2A.1.5.1 | Collect_Brightness_Temps | | N | | |
| L2A.2.1.1 | Determine_Composites | N | | | |
| L2A.2.1.2 | Determine_Slice_Quality | N | | | |
| L2A.2.1.3 | Get_Composite_Usability | N | | | |
| L2A.2.1.4 | Composite_Slices | N | | | |
| L2A.2.1.5 | Composite_Sigma0_Parameters_Sig | N | | | |
| L2A.2.2.1 | Calculate_Tb_Statistics | | N | | |
| L2A.3.1.1 | Flag_Read_Map | N | | | |
| L2A.3.1.2 | Flag_By_Mask | N | | | |
| Level 2B Algorithms | | | | | |
| L2B.1.1.1 | Execute_Wind_Algorithms | N | | | |
| L2B.1.2.1 | Prepare_WVC | N | | | |
| L2B.1.2.2 | Filter_WVC | N | | | |
| L2B.1.2.3 | Calculate_WVC_Centroid | N | | | |
| L2B.1.2.4 | Calculate_WVC_Brightness_Temp | | N | | |
| L2B.1.2.5 | Set_Winds_Flags | N | | | |
| L2B.2.1.1 | Retrieve_Winds | N | | | |
| L2B.2.1.2 | Calculate_Initial_WVC_Solutions | N | | | |
| L2B.2.1.3 | Optimize_Wind_Solutions | N | | | |
| L2B.2.1.4 | Rank_Wind_Solutions | N | | | |
| L2B.2.2.1 | Evaluate_Objective_Function | N | | | |
| L2B.2.2.2 | Run_Model_Function | N | | | |
| L2B.2.2.3 | Initialize_Model_Function | N | | | |

| | | | | | |
|----------------------|--|---|---|----|------------------------|
| L2B.2.2.4 | Initialize_Interpolation_Coefficients | N | | | |
| L2B.2.2.5 | Initialize_Working_Table | N | | | |
| L2B.2.2.6 | Evaluate_Model_Function | N | | | |
| L2B.3.1.1 | Determine_Rain_Flag | | N | | |
| L2B.3.1.2 | Calculate_NOF_Rain_Index | | N | | |
| L2B.3.1.3 | Calculate_Transmittance | | N | | |
| L2B.3.2.1 | Filter_Rain_Flag_Spatially | | N | | |
| L2B.4.1.1 | Execute_Ambiguity_Removal | N | | MS | see Multi-pass version |
| L2B.4.1.2.1 | Select_Ambiguity | N | | | |
| L2B.4.1.2.2 | Select_Nudge | N | | | |
| L2B.4.1.2.3 | Determine_NWP_Nudge | N | | | |
| L2B.4.1.3 | Initialize_Working_Arrays | N | | MS | see Multi-pass version |
| L2B.4.1.4 | Calculate_Median_Filter | N | | | |
| L2B.4.1.5 | Replace_Ambiguity_Buffers | N | | | |
| L2B.4.1.1(MP) | Execute_Ambiguity_Removal (multi-pass) | | | N | |
| L2B.4.1.3(MP) | Initialize_Working_Arrays | | | N | |
| L2B.5.1.1 | Convert_Objective_Function to PDF | | N | | |
| L2B.5.1.2 | Build_Dir_Ranges | | N | | |
| L2B.5.1.3 | Correct_Selected_Vectors | | N | | |
| L2B.5.1.4 | DIR_Median_Filter | | N | | |
| L2B.5.2.1 | Thresholded_Nudging | | N | | |
| L2B.2.1.1 (DIRTH) | Retrieve_Winds (DIRTH) | | N | | |
| L2B.2.1.2 (DIRTH) | Calculate_Initial_WVC_ Solution (DIRTH) | | N | | |

SeaWinds Spacecraft Location and Nadir Location Algorithm

MODULE L1A.1.0
MODULE L1B.1.0

ALGORITHM SPECIFICATIONS

AUTHOR: R. Scott Dunbar
VERSION: 2.0
DATE: October, 1999

SeaWinds Spacecraft Location and Nadir Location Algorithm

I. Module Overview

In processing SeaWinds telemetry data frames to Level 1A, we are required to determine the precise spacecraft state vector (position and velocity) at a given frame time. In addition, the first step in locating SeaWinds pulse and slice footprints at Level 1B is to determine the precise location and velocity of the spacecraft, and the Earth location of its nadir point, at a given pulse time. The spacecraft ephemeris, which for QuikSCAT is obtained from the onboard GPS processor and embedded in the science housekeeping (HK2) data, and for ADEOS-II is in the Platform Correction Data (PCD) of the SeaWinds telemetry packets, is interpolated at the given time to obtain the position and velocity data required for the L1A product. In the L1B algorithm processing, the state vector data is passed to the SWS Geometry computations (L1B.2.0). This algorithm module that performs these tasks is common to both the L1A and L1B processing.

The nominal rate of ephemeris data for SeaWinds depends on the source of the data. The ADEOS-2 ELMD data is expected to provide state vectors at 1-minute intervals, and the ADEOS-2 PCD data embedded in the telemetry will provide data at 8-second intervals. The QuikSCAT GPS processor data in the HK2 is nominally at 2-second intervals. However, in the latter case, the science and housekeeping data packet generation process in the spacecraft onboard computer has a systematic timing problem in its FIFO buffer, the result of which is a periodic loss of one or more HK2 packets. This means that we can no longer rely on an ephemeris interpolator that depends on equally-spaced data, such as the previous NSCAT-derived version of the interpolation algorithm.

Given the variety of possible inputs and the potential for non-uniform spacing of the data in time, the revised ephemeris interpolation algorithm fits a cubic spline to each of the six state coordinates (position and velocity vectors) of the spacecraft over the full time span of the data being processed. The cubic spline was selected for its robustness, ability to handle small and moderate-sized data gaps, and a further ability to compute an error estimate for each interpolation that can be used to flag the data. This application of the spline algorithm to the SeaWinds ephemeris interpolation problem was first proposed and described in [1].

Once the spacecraft position vector is known, the nadir location algorithm is used to determine the geodetic coordinates of the spacecraft nadir and the altitude. This algorithm assumes an ellipsoidal Earth, and so the computation of the geodetic latitude and the altitude is a simultaneous, iterative solution for both quantities. This algorithm remains unchanged from the original (V1.0).

The main outputs of this algorithm needed for subsequent processing are the spacecraft position and velocity vectors (in the Earth-rotating Cartesian frame), the nadir latitude and longitude, and the spacecraft altitude.

Performance Requirements

The accuracy of the interpolation technique used mainly depends on the time interval between ephemeris entries. Assuming an acceptable 1-axis positional error of 100 meters, a time interval (or data gap) up to 200-300 seconds can be handled. Beyond a 300 second interval, the oscillation of the spline interpolator near the edge of the data (or data gap) makes the ephemeris accuracy unacceptable. Therefore it is necessary to buffer ephemeris data extending at least 5 minutes beyond the ends of the particular Level 0 (for the L1A processor) or L1A (for the L1B processor) data being processed. However, since the QuikSCAT processing system already uses a much larger span of ephemeris data for a given time range of scatterometer data (e.g. previous, current, and next files), this should not incur problems.

Processing speed and accuracy were tested during the development of this algorithm using a simulated one-rev Level 1A data file. Timing test results are given in [1].

II. Functional Flow Description

The processing for this algorithm is divided into an initialization stage, performed at the beginning of the processing, and the actual interpolation stage, which is called as required by the main level processor. In the L1B processing, these are followed by the nadir location algorithm. These stages include the following functions:

(1.1) Initialization stage:

- 1.1.1 Initialize Ephemeris Interpolation
- 1.1.2 Read and Buffer Ephemeris Data
- 1.1.3 Compute Spline Coefficients

(1.2) Interpolation stage:

- 1.2.1 Interpolate State Vector
- 1.2.2 Compute Spline

(1.3) Nadir location (L1B only)

- 1.3.1 NADIR

REFERENCE

- [1] Dunbar, R. S. (1998) An Alternative Ephemeris Interpolation Algorithm for QuikSCAT, JPL IOM 3349-98-015, November 20, 1998.

SeaWinds Algorithm Specification

TITLE: Initialize Ephemeris Interpolation
SUBMODULE: Initialize Ephemeris Interpolation
MODULE: SWS Spacecraft Location and Nadir Location
CODE: L1A.1.1.1/ L1B.1.1.1
VERSION: 2.0
DATE: February 22, 1999
AUTHOR: R. Scott Dunbar
SUBROUTINE: Initialize_Ephemeris.F
LANGUAGE: FORTRAN
HERITAGE:

L1A.1.1.1/L1B.1.1.1 Initialize_Ephemeris

PURPOSE

Controlling routine for the initialization of the ephemeris interpolation algorithm.

REQUIREMENTS

Control the setup of the cubic spline ephemeris interpolation using the ephemeris data in the ephemeris buffer.

BACKGROUND

The cubic spline interpolator [1] is one of many alternative algorithms we could choose to solve our ephemeris problem. The cubic spline algorithm uses an interpolation formula of the form:

$$y = Ay_j + By_{j+1} + Cy''_j + Dy''_{j+1}$$

where

$$A = (x_{j+1} - x)/(x_{j+1} - x_j)$$

$$B = 1 - A = (x - x_j)/(x_{j+1} - x_j)$$

$$C = (1/6) (A^3 - A) (x_{j+1} - x_j)^2$$

$$D = (1/6) (B^3 - B) (x_{j+1} - x_j)^2$$

The y_j are the tabulated values of the function that are to be interpolated, and the second derivatives y''_j are computed by demanding that the formula should return the exact tabular

values y_j given $x = x_j$, and that the first derivatives should be continuous. These constraints lead to a tridiagonal system of equations that can be solved for the y''_j in $O(N)$ operations, where N is the number of tabulated points being considered in the spline fit. There is no assumption in the formulation of the cubic spline as to the regularity of the tabulated values – equal spacing is not required. In fact, the spline is often used to produce an equally-spaced array for other applications that require it, such as fast Fourier transforms.

In the present problem, we have six independent spline fits that are required, one for each state vector coordinate $(X, Y, Z, V_x, V_y, V_z)(t)$ in the ephemeris. The fits, consisting of computing the y''_j values to match the spline constraints, are done once for each ephemeris coordinate after reading the ephemeris data (at the beginning of processing), and then the interpolation equation is applied for each coordinate for any time inside the time span of the data. Note that the cubic spline is *not* a good extrapolator; significant oscillation of the interpolated values occur within about 4-5 time intervals of the ends of the fit, and much worse oscillation occurs outside the time span. However, since the QuikSCAT processing system already uses a much larger span of ephemeris data for a given time range of scatterometer data (e.g. previous, current, and next files), this should not incur problems.

INPUTS

| | | |
|--------------------|-----|--|
| neph_total | I*4 | Number of unique ephemeris data points (times) actually read from the ephemeris file(s). |
| ephTime | R*8 | Array of [NPTS] times of ephemeris data, expressed in seconds from January 1, 1993 |
| xPos,yPos, zPos | R*8 | Arrays of [NPTS] spacecraft position vector components, in meters |
| xVel,yVel, zVel | R*8 | Arrays of [NPTS] spacecraft velocity vector components, in m/sec |

OUTPUTS

| | | |
|-------|-----|---|
| Deriv | R*8 | Array of [NPTS,6] values of the second derivatives of the six spline functions of the state vector components, computed at each ephemeris data point. |
|-------|-----|---|

PROCESSING

- Step 1. It is assumed that the ephemeris data have been previously read into memory via module 1.1.2 (Read_Ephemeris).
- Step 2. Call module 1.1.3 (Compute Spline Coefficients) once for each state vector coordinate (six times in all).
- Step 3. Return to calling program (e.g. Level processor).

AUXILIARY DATA

| | | |
|----------|-----|---|
| NPTS | I*4 | Maximum number of ephemeris data points (times) that can be stored in a given run; sizes the arrays ephTime, xPos, yPos, zPos, xVel, yVel, and zVel. |
| yp1, ypn | R*8 | “Flag” values for the first derivatives on the boundaries, setting the “natural” cubic spline boundary conditions of $y''_1 = y''_n = 0$. Values set to /1.d31/. |

REFERENCES

- [1] Press, Flannery, Teukolsky, Vetterling (1986) “Cubic Spline Interpolation”, Numerical Recipes, section 3.3, pp. 86-89, Cambridge University Press, 1986.

SeaWinds Algorithm Specification

TITLE: Read and Buffer Ephemeris Data
SUBMODULE: Initialize Ephemeris Interpolation
MODULE: SWS Spacecraft Location and Nadir Location
CODE: L1A.1.1.2/ L1B.1.1.2
VERSION: 2.0
DATE: February 22, 1999
AUTHOR: R. Scott Dunbar
SUBROUTINE: Read_Ephemeris.F
LANGUAGE: FORTRAN
HERITAGE:

L1A.1.1.2/L1B.1.1.2 Read_Ephemeris

PURPOSE

Read ephemeris times and state vectors from a specified set of files, creating a buffer of unique ephemeris data points for use in the ephemeris interpolation algorithm.

REQUIREMENTS

Times of ephemeris entries must be unique. Overlapping data between data files must be treated such that only one set of data from the overlap remains in the buffer.

BACKGROUND

The ephemeris data file contains the set of spacecraft state vectors (position and velocity in Earth-rotating cartesian coordinates) and their associated times (in seconds from 0 UTC, January 1, 1993), as generated by the onboard GPS processor (for QuikSCAT) or from ground-based orbit propagation (QuikSCAT backup and ADEOS-2). For QuikSCAT the files are created from the science housekeeping (HK2) telemetry by the SeaPAC HK2 preprocessor. The format and content of the files is given in the Ephemeris Data file SIS [1]. The same format and content will be maintained for the ephemeris data regardless of the data source (only the filenames and header metadata will be different).

INPUTS

| | | |
|----------|--------|--|
| Ephfiles | STRING | Array of [NFILES] ephemeris file names from which the ephemeris data are to be read into the buffer. The sequence of files is assumed to be in time-order. |
|----------|--------|--|

OUTPUTS

| | | |
|--------------------|-----|--|
| neph_total | I*4 | Number of unique ephemeris data points (times) actually read from the ephemeris file(s). |
| ephTime | R*8 | Array of [NPTS] times of ephemeris data, expressed in seconds from January 1, 1993 |
| xPos,yPos, zPos | R*8 | Arrays of [NPTS] spacecraft position vector components, in meters |
| xVel,yVel, zVel | R*8 | Arrays of [NPTS] spacecraft velocity vector components, in m/sec |

PROCESSING

Step 1. Initialize the value of the variable LAST_TIME to 0.d0.

For each file in the list of filenames:

Step 2. Open the file and determine the number of data records in the file from the header metadata.

Step 3. Read each data record sequentially.

Step 4. Compare the value of the ephemeris time in the data record to the value of LAST_TIME; if the ephemeris time is greater, increment NEPH_TOTAL, and copy the time and state vector component values to the buffer arrays; if the ephemeris time is less than or equal to LAST_TIME, go back to step 3 and read another record without storing the current record in the buffer.

After all of the files in the list have been read, return the value of NEPH_TOTAL and the buffer arrays ephTime, xPos, yPos, zPos, xVel, yVel, and zVel to the calling program.

AUXILIARY DATA

| | | |
|------|-----|--|
| NPTS | I*4 | Maximum number of ephemeris data points (times) that can be stored in a given run; sizes the arrays ephTime, xPos, yPos, zPos, xVel, yVel, and zVel. |
|------|-----|--|

INTERNAL VARIABLES

| | | |
|-----------|-----|--|
| Last_time | R*8 | Time read for the previous ephemeris data point. |
|-----------|-----|--|

COMMENTS

It is optional whether one adopts the “first data is best” which is described here, or a “last data is best” philosophy when implementing the overlap elimination, so long as the uniqueness

requirement is satisfied.

REFERENCES

- [1] Merida, Sofia (1998). System Interface Specifications for SeaWinds Ephemeris Data, JPL document 686-xxx.

SeaWinds Algorithm Specification

TITLE: Compute Spline Coefficients
SUBMODULE: Initialize Ephemeris Interpolation
MODULE: SWS Spacecraft Location and Nadir Location
CODE: L1A.1.1.3/L1B.1.1.3
VERSION: 2.0
DATE: February 22, 1999
AUTHOR: R. Scott Dunbar
SUBROUTINE: SPLINE.F
LANGUAGE: FORTRAN
HERITAGE: *Numerical Recipes*

L1A.1.1.3/L1B.1.1.3 Spline

PURPOSE

Compute the second derivatives $y''(i)$ of the cubic spline interpolating function $y = y(x)$ for a given set of N data points $(x, y)_i$ ($i = 1, N$).

BACKGROUND

The cubic spline interpolator [1] is applicable to unevenly-spaced data, such as that which we expect in the QuikSCAT ephemeris. The algorithm used here is taken from *Numerical Recipes*, and has been only slightly modified from the original to handle double precision data arrays. This algorithm embeds the tridiagonal solution for the second derivative values directly into the code. The second derivatives of y at the data points are required for the later interpolation step (1.2.2).

INPUTS

| | | |
|----------|-----|--|
| N | I*4 | Number of data points in the X and Y arrays |
| X | R*8 | Array of abscissae (times), max dimension $NMAX \geq N$ |
| Y | R*8 | Array of ordinates (state coordinates), max dimension $NMAX \geq N$ |
| YP1, YPN | R*8 | Boundary values of the first derivatives; values are set to 1.d31 to specify a <i>natural</i> cubic spline |

OUTPUTS

| | | |
|----|-----|---|
| Y2 | R*8 | Array of computed second derivatives, max dimension $NMAX \geq N$ |
|----|-----|---|

PROCESSING

Step 1. For natural cubic spline ($YP1 > .99d30$), set the boundary values

$$Y2(1)=0 \text{ and } U(1)=0.$$

Step 2. For points $I=2$ to $N-1$ compute the following:

$$SIG = (X(I) - X(I-1))/(X(I+1) - X(I-1))$$

$$P = SIG*Y2(I-1) + 2.$$

$$Y2(I) = (SIG - 1.)/P$$

$$U(I) = (6.*((Y(I+1) - Y(I))/(X(I+1) - X(I)) - (Y(I) - Y(I-1)) / (X(I) - X(I-1))) / (X(I+1) - X(I-1)) - SIG*U(I-1))/P$$

Step 3. For natural cubic spline ($YPN > .99d30$), set the boundary value $Y2(N) = 0$.

Step 4. Compute the final $Y2$ array elements using the recursion:

$$Y2(I) = Y2(I)*Y2(I+1) + U(I) \text{ for } I=N-1,1,-1$$

Step 5. Return the array $Y2$ to the calling routine.

INTERNAL VARIABLES

| | | |
|-----|-----|--|
| U | R*8 | Auxiliary function array used in the $Y2$ computations, max dimension $NMAX \geq N$ |
| SIG | R*8 | Temporary variable used in the tridiagonal solution for $Y2(I)$ |
| P | R*8 | Temporary variable used in the tridiagonal solution for $Y2(I)$ |

REFERENCE

- [1] Press, Flannery, Teukolsky, Vetterling (1986) "Cubic Spline Interpolation", Numerical Recipes, section 3.3, pp. 86-89, Cambridge University Press, 1986.

SeaWinds Algorithm Specification

TITLE: Interpolate State Vector
SUBMODULE: Ephemeris Interpolation
MODULE: SWS Spacecraft Location and Nadir Location
CODE: L1A.1.2.1/L1B.1.2.1
VERSION: 2.0
DATE: February 22, 1999
AUTHOR: R. Scott Dunbar
SUBROUTINE: Interpolate_Ephemeris.F
LANGUAGE: FORTRAN
HERITAGE:

L1A.1.2.1/L1B.1.2.1 Interpolate_Ephemeris

PURPOSE

Controlling routine for interpolation of the ephemeris for any given time in the time range of the data.

BACKGROUND

In the present problem, we have six independent spline fits that are required, one for each state vector coordinate (X, Y, Z, V_x, V_y, V_z)(t) in the ephemeris. The fits, consisting of computing the y''_j values to match the spline constraints, are done once for each ephemeris coordinate after reading the ephemeris data (at the beginning of processing), and then the interpolation equation is applied for each coordinate for any time inside the time span of the data. Note that the cubic spline is *not* a good extrapolator; significant oscillation of the interpolated values occur within about 4-5 time intervals of the ends of the fit, and much worse oscillation occurs outside the time span. However, since the QuikSCAT processing system already uses a much larger span of ephemeris data for a given time range of scatterometer data (e.g. previous, current, and next files), this should not incur problems.

The error bound on a cubic spline interpolation is given in [1] by:

$$\varepsilon \sim (1/4!) \max |f^{(4)}(x)| (\Delta x/2)^4$$

where $\Delta x = x_{j+1} - x_j$ and $f^{(4)}(x)$ is the fourth derivative of the tabulated function. The maximum interpolation error occurs at the mid-point of the interpolation interval. For the ephemeris problem, all of the state coordinates are sinusoidal functions with the primary variation being due to the orbital frequency, e.g.:

$$X \sim A \sin(\omega t + \phi),$$

so that

$$\max |X^{(4)}(t)| \sim A \omega^4.$$

In terms of the semimajor axis a of the orbit (about 7.2×10^6 meters for QuikSCAT) and the mass parameter of the Earth $\mu = 3.986032 \times 10^{14} \text{ m}^3/\text{sec}^2$ the error bound can be written as:

$$\varepsilon \sim (1/384) \mu^2 (\Delta t)^4 / a^5 = 2.1 \times 10^{-8} (\Delta t)^4 \text{ meter.}$$

Note that this is supposed to be an upper bound, yet for time intervals of up to 200 to 300 seconds the resulting expected position error is much less than the expected uncertainty of the GPS ephemeris. This simple formula, parametrized only by the time interval over which a particular interpolation is performed, provides the means by which the derived ephemeris values used in the processing can be quality-controlled, and flagged if needed.

INPUTS

From calling program (e.g. Level processor):

| | | |
|--------------------|-----|---|
| time | R*8 | Time, expressed in seconds from January 1, 1993, at which the ephemeris is to be interpolated |
| neph_total | I*4 | Number of unique ephemeris data points (times) actually read from the ephemeris file(s). |
| ephTime | R*8 | Array of [NPTS] times of ephemeris data, expressed in seconds from January 1, 1993 |
| xPos,yPos, zPos | R*8 | Arrays of [NPTS] spacecraft position vector components, in meters |
| xVel,yVel, zVel | R*8 | Arrays of [NPTS] spacecraft velocity vector components, in m/sec |
| deriv | R*8 | Array of [NPTS,6] values of the second derivatives of the six spline functions of the state vector components, computed at each ephemeris data point. |

From call(s) to module 1.2.2 (Compute Spline Interpolator):

| | | |
|----|-----|--|
| dt | R*8 | Time interval between data points in which the requested time falls; used to compute error estimate. |
|----|-----|--|

OUTPUTS

| | | |
|---------|-----|---|
| cpos | R*8 | Array of interpolated spacecraft position coordinates (X,Y,Z = 1,2,3) |
| cvel | R*8 | Array of interpolated spacecraft velocity coordinates (X,Y,Z = 1,2,3) |
| err_est | R*8 | Value of the error estimator ε , expressed in meters |

PROCESSING

- Step 1. Call module 1.2.2 (Compute Spline Interpolator) once for each state vector coordinate (six times in all). These calls fill the interpolated state vector arrays **cpos** and **cvel**.
- Step 2. Using the value **dt** of the data time interval, compute the error estimate **err_est**.
- Step 3. Return **cpos**, **cvel**, and **err_est** to the calling program.

AUXILIARY DATA

The following are needed in the computation of the interpolation error estimate:

aorb = 7.2d6 approximate value of the orbit semimajor axis (meters)
xmu = 3.986005d14 Earth gravitational parameter (m**3/sec**2)

REFERENCES

- [1] Burden, R. L., J. D. Faires, and A. C. Reynolds, Numerical Analysis, pp. 107-119 (error formula is on p. 119).

SeaWinds Algorithm Specification

TITLE: Compute Spline Interpolator
SUBMODULE: Spacecraft Ephemeris Interpolation
MODULE: SWS Spacecraft Location and Nadir Location
CODE: L1B.1.2.2
VERSION: 2.0
DATE: October 19, 2001
AUTHOR: R. Scott Dunbar
SUBROUTINE: SPLINT.F
LANGUAGE: FORTRAN
HERITAGE: *Numerical Recipes*

L1B.1.2.2 Spline Interpolator

PURPOSE

Compute the cubic spline interpolator to obtain $y = y(x)$ at a given value of x . This is a general-purpose routine for the cubic spline calculation taken from *Numerical Recipes* [1].

REQUIREMENTS

This algorithm shall contribute errors of no more than 100 m in the position components and 0.1 m/sec in the velocity components of the spacecraft state vector as compared to an "exact" ephemeris computation at the given time.

BACKGROUND

The cubic spline interpolator [1] is one of many alternative algorithms we could choose to solve our ephemeris problem. The cubic spline algorithm uses an interpolation formula of the form:

$$y = Ay_j + By_{j+1} + Cy''_j + Dy''_{j+1}$$

where

$$A = (x_{j+1} - x)/(x_{j+1} - x_j)$$

$$B = 1 - A = (x - x_j)/(x_{j+1} - x_j)$$

$$C = (1/6) (A^3 - A) (x_{j+1} - x_j)^2$$

$$D = (1/6) (B^3 - B) (x_{j+1} - x_j)^2$$

The y_j are the tabulated values of the function that are to be interpolated, and the second derivatives y''_j are computed by demanding that the formula should return the exact tabular values y_j given $x = x_j$, and that the first derivatives should be continuous. The interpolator is called once for each state vector coordinate at the given time.

INPUTS

| | | |
|-----|-----|---|
| N | I*4 | Number of data points in the X and Y arrays |
| XA | R*8 | Array of abscissae (times), max dimension $NMAX \geq N$ |
| YA | R*8 | Array of ordinates (state coordinates), max dimension $NMAX \geq N$ |
| Y2A | R*8 | Array of computed second derivatives, max dimension $NMAX \geq N$ |
| X | R*8 | Time at which the interpolated value is to be computed |

OUTPUTS

| | | |
|---|-----|---|
| Y | R*8 | Cubic spline interpolated value of YA at time X |
| H | R*8 | Length of the time interval (ΔX) in which the value of X falls (needed for computing the interpolation error) |

PROCESSING

Step 1. Determine the bounding data points KLO and KHI of X in the XA array by bisection.

Step 2. Compute the value of $H = XA(KHI) - XA(KLO)$ = time interval in seconds.

Step 3. Compute the cubic spline interpolated value of Y as follows:

$$A = (XA(KHI) - X)/H$$

$$B = (X - XA(KLO))/H$$

$$Y = A*YA(KLO) + B*YA(KHI) + ((A**3 - A)*Y2A(KLO) + (B**3 - B)*Y2A(KHI))*(H**2)/6.$$

INTERNAL VARIABLES

| | | |
|------|-----|--|
| A, B | R*8 | Temporary variables used in the computation of the spline interpolator |
|------|-----|--|

REFERENCE

- [1] Press, Flannery, Teukolsky, Vetterling (1986) "Cubic Spline Interpolation", Numerical Recipes, section 3.3, pp. 86-89, Cambridge University Press, 1986.

SeaWinds Algorithm Specification

TITLE: Locate Spacecraft Nadir
SUBMODULE: Spacecraft Nadir Location
MODULE: SWS Spacecraft Location and Nadir Location
CODE: L1B.1.3.1
VERSION: 2.0
DATE: February 22, 1999
AUTHOR: R. Scott Dunbar
SUBROUTINE: NADIR
LANGUAGE: FORTRAN
HERITAGE: NSCAT ATB (nadir.F)

L1B.1.3.1 Nadir

PURPOSE

Compute geodetic location of spacecraft nadir point from knowledge of the spacecraft location at a given frame time.

REQUIREMENTS

The error introduced by the nadir location algorithm in the output quantities, given the spacecraft position coordinates, shall be no larger than 1 part in 10^4 .

BACKGROUND

The geodetic location of the spacecraft nadir point and the spacecraft altitude are required for subsequent Level 1B processing. These quantities may be computed for a given frame time once the cartesian spacecraft coordinates have been determined from the ephemeris.

The algorithm described here assumes an ellipsoidal Earth with equatorial radius R and flattening factor f . Refer to Figure 3 for a representation of the geometry of this problem. The spacecraft coordinates are given by (x,y,z) , and the nadir coordinates to be determined are the longitude λ , the geodetic latitude ϕ , and the altitude H . Computation of the nadir longitude is straightforward, but ϕ and H must be solved for simultaneously by an iterative procedure because of the ellipsoid. It is also useful to define the (constant) flattening factors

$$f_1 = 1 - f \tag{1}$$

$$f_2 = f(2 - f)$$

to simplify the formulation. We also compute the distances

$$\rho = [x^2 + y^2 + z^2]^{1/2} \quad (2)$$

and

$$r = [x^2 + y^2]^{1/2} \quad (3)$$

The nadir longitude is found from

$$L = \tan^{-1}(y/x) \quad (4)$$

Trial values of ϕ and H are then determined by

$$\phi_0 = \sin^{-1}(z/\rho) \quad (5)$$

and

$$H_0 = \rho - R(1 - f \sin^2 \phi) \quad (6)$$

The iterative solution for ϕ and H begins by computing the factors

$$g_0 = R [1 - f_2 \sin^2 \phi]^{-1/2} \quad (7)$$

$$g_1 = g_0 + H \quad (8)$$

$$g_2 = g_0 (1 - f)^2 + H \quad (9)$$

Then corrections to r, z, H, and ϕ can be formed:

$$dr = r - g_1 \cos \phi \quad (10)$$

$$dz = z - g_2 \sin \phi \quad (11)$$

$$dH = dr \cos \phi + dz \sin \phi \quad (12)$$

$$d\phi = (dz \cos \phi - dr \sin \phi)/(R + H + dH) \quad (13)$$

The new values of H and ϕ are thus

$$\phi = \phi + d\phi \quad (14)$$

and

$$H = H + dH \quad (15)$$

The process has converged if, given a tolerance ϵ :

$$|d\phi| \leq \epsilon$$

and

$$|dH|/(R + H) \leq \epsilon \quad (16)$$

If these relations are not satisfied, we insert the new values of j and H from (14) and (15) into the computation starting at equation (7) and compute new corrections. This process converges rapidly, although for safety we may set a limit on the number of iterations which, if exceeded, returns the current values with a "no convergence" flag. The final outputs are from equations (4), (14), and (15), plus the logical convergence flag.

INPUTS

| | | |
|--------|-----|--|
| XYZ(3) | R*8 | Cartesian coordinates of the spacecraft, determined in EPHINT (L1B.1.1.1). |
|--------|-----|--|

OUTPUTS

| | | |
|-------|-----|--|
| GLAT | R*8 | Geodetic latitude of spacecraft nadir, degrees. |
| ELON | R*8 | Longitude of spacecraft nadir point, degrees. |
| HT | R*8 | Spacecraft altitude in kilometers. |
| ICONV | L*4 | Convergence flag for the iteration; .true. if converged, .false. if not. |

PROCESSING

Step 1. Initialize convergence flag $ICONV = .true.$, and compute the flattening factors and $r = \sqrt{x*x + y*y}$ needed in later computations.

Step 2. Check that $XYZ(1)$ is not zero to avoid arctangent problems in computing nadir longitude; if it is, set $x = 10^{-11}$.

Step 3. Compute nadir longitude λ (make sure that $ELON > 0$), and convert to degrees:

$$\lambda = ELON = \text{datan2}(y,x) * \text{rtd}$$

Step 4. Compute trial values of RHO , $GLAT$, and HT (eqs. 2,5,6) to start off the iteration, and initialize an iteration counter.

Step 5. Compute corrections to $GLAT$ and HT using eqs. 7-15.

Step 6. Check the size of the corrections against the error tolerance to test convergence, and check that the number of iterations has not exceeded the maximum. If not converged, go back to step (5). If the iteration limit has been exceeded, set $ICONV = .false.$ and return to EPHINT.

Step 7. If convergence has been achieved, convert $GLAT$ to degrees (HT is already done), and return to EPHINT with $ICONV = .true.$

AUXILIARY DATA

| | | | |
|-------|-----|--|----------------------|
| AE | R*8 | Earth equatorial radius, kilometers | = 6378.197 |
| FLAT | R*8 | Earth ellipsoid flattening factor | = 3.352810665d-3 |
| PI | R*8 | Circumference/diameter ratio | = 3.14159265358979 |
| TWOPI | R*8 | 2*PI | |
| DTR | R*8 | Degree-to-radian conversion | = 0.0174532925199433 |
| RTD | R*8 | Radian-to-degree conversion | = 57.2957795130823 |
| TOL | R*8 | Convergence tolerance (machine precision limit) | = 1.0d-14 |
| MAX | I*4 | Iteration limit | = 10 |

INTERNAL VARIABLES

| | | |
|-------|-----|---|
| ITER | I*4 | Iteration counter |
| DR | R*8 | Correction to r (eq. 10) |
| DZ | R*8 | Correction to z (eq. 11) |
| DHT | R*8 | Correction to H (eq. 12) |
| DLATR | R*8 | Correction to ϕ (radians) (eq. 13) |

COMMENTS

Double precision computations (variables and functions) should be used throughout.

REFERENCES

- [1] Escobal, P. R. Methods of Orbit Determination, Krieger Publishing Co., Inc., 1976, pp. 23-28.
- [2] Science Algorithm Specifications for the NASA Scatterometer Project, Vol. 2.

SeaWinds Geometry Algorithms

Module L1B.2.0

ALGORITHM SPECIFICATIONS

| | |
|-------------------|--|
| AUTHOR(s): | S. Vincent Hsiao Anzhen Zhang |
| VERSION: | 2.0 |
| DATE: | October 31, 1999 |

SeaWinds Geometry Algorithms

MODULE L1B.2.0

I. Module Overview

The SeaWinds Geometry Algorithms Module encompasses all calculations related to the determination of backscatter footprint location and other geometric parameters needed for all subsequent algorithm processing. This module is subdivided into six principal submodules:

- L1B.2.1 Geometry Algorithm Interface from 1A to 1B Processor
- L1B.2.2 General Geometry (coordinate systems and transformations)
- L1B.2.3 Antenna Geometry (antenna pointing)
- L1B.2.4 Cell Location and Geometry (footprint location, range, etc.)

These submodules are described in the following sections.

GEOMETRY ALGORITHM INTERFACE FROM 1A TO 1B PROCESSOR(L1B.2.1)

This is the driver part of the Geometry algorithm. It reads the Level 1A data file, calls the other submodules, and writes the Level 1B data file.

GENERAL GEOMETRY (L1B.2.2)

This submodule specifies the coordinate systems used in SeaWinds geometry computations and the transformations between coordinate systems. Geometry calculations related to spacecraft (s/c) position are also included in this module. The outputs are used in the Antenna Geometry and Cell Geometry algorithm submodules.

There are three coordinate systems used in SeaWinds geometry computations (see Figure 1). The first is an Earth-centered, Earth-fixed coordinate system. The second is a s/c-centered local coordinate system whose orientation is determined by the s/c position and inertial velocity vectors. The third is a s/c-centered, s/c-fixed spacecraft body coordinate system. Details and transformations of the coordinate systems can be found later in this specification.

The following geometry computations are performed in this module:

1. The geocentric s/c position vector;
2. The orientation of local coordinate system in the geocentric system;
3. The matrix for the coordinate transformation due to roll, pitch, and yaw of the s/c.

ANTENNA GEOMETRY (L1B.2.3)

This submodule performs all geometric calculations related to antenna pointing. In this submodule the antenna maximum gain direction is computed by considering the roll, pitch, and yaw angles of s/c. The direction vector from the spacecraft to the egg center is also calculated in

this submodule. The outputs are used in the Cell Geometry submodule to locate the egg center on the Earth's surface.

CELL LOCATION AND GEOMETRY (L1B.2.4)

This submodule uses the outputs of the General Geometry and Antenna Geometry algorithms to determine the location and geometry of both eggs and slices. The outputs of this submodule are used in surface flag detection (L2A.1.4), the σ_0 and K_p computation (L1B.3.0), subtrack binning (L2A.1.3), and wind retrieval (L2B.2.0).

The direction from the spacecraft to the egg center computed in Antenna Geometry is used to compute the position vector of the egg center. The longitude and geocentric latitude of the egg center are then calculated.

The X values for both eggs and slices are computed from the X-factor table. The elevation and azimuth angles, with respect to the egg center, of all the slices are computed from the slice location table. The slice locations are then calculated.

Once the locations of the eggs and slices are known, the following quantities can be computed:

- Geodetic latitude and longitude for grouping and for land and ice flagging;
- Azimuth angle and incidence angle, for wind retrieval.

II. Functional Flow Description

The submodule Geometry Algorithm Interface (L1B.2.1) is first invoked in the Level 1B processing. Submodules L1B.2.2 to 2.5 are then called from and controlled by the Geometry Algorithm Interface.

GENERAL GEOMETRY:

L1B.2.2.1 - Local Coordinate System - (COMPUTE_LOCAL_COORD) determines the orientation of the local coordinate system;

L1B.2.2.2 - Attitude Matrix - (COMPUTE_ATTITUDE_ROTATN_MATRIX) calculates the elements of the matrix for coordination transformation due to roll, pitch, and yaw;

L1B.2.2.3 - Spacecraft Body to Local Coordinate Transformation - (CONVERT_SC_TO_LOCAL) transforms s/c body coordinates to local coordinates including attitude adjustments. Used in Antenna Geometry computations to transform the maximum gain directions;

L1B.2.2.4 - Geocentric-Local Transformation - (CONVERT_BETWEEN_RECT_LOCAL) converts Earth-fixed s/c velocity from the geocentric coordinate system to the s/c-centered local coordinate system;

L1B.2.2.5 - Rectangular to Longitude-Latitude Transformation - (CONVERT_RECT_TO_GEO) Converts cartesian coordinates to geocentric latitude and longitude;

L1B.2.2.6 - Geocentric to Geodetic Latitude Transformation - (COMPUTE_GEODETTIC_LAT) uses the adopted Earth ellipsoid model to convert geocentric latitude to geodetic latitude;

L1B.2.2.7 - Convert from Geocentric to Geographic Coordinates - (CONVERT_GEOCEN_GEOGRAPH) computes the transformation matrix from the geocentric to (N,E,D) (North-East-Down) coordinate systems;

ANTENNA GEOMETRY:

L1B.2.3.1 - Boresight - (COMPUTE_MAX_GAIN_DIR) calculates the antenna maximum gain direction in s/c body coordinate system;

L1B.2.3.2 - Calculation of Measurement Time - (DETERMINE MEASUREMENT_TIME) Determine the measurement time for every measurement.

CELL LOCATION AND GEOMETRY:

L1B.2.4.1 - Locate Cell Center - (LOCATE_CELL_ON_EARTH) locates the footprint center on Earth surface in rectangular coordinates, which is then transformed by CONVERT_RECT_TO_GEO to geocentric longitudes and latitudes;

L1B.2.4.2 - Incidence Angle - (COMPUTE_INCIDENCE_ANGLE) calculates the incidence angle;

L1B.2.4.4 - Cell Azimuth - (COMPUTE_CELL_AZIMUTH_ANGLE) calculates the footprint azimuth angle from north;

L1B.2.4.5 - Range Tracking and Doppler Shift - (DOPPLER_SHIFT_RANGE_TRACKING) duplicates the CDS calculations of Doppler shift and range tracking;

L1B.2.4.6 - X-factor and Slice Location - (COMPUTE_X_LOC) calculates the X factor values and slice locations from tables.

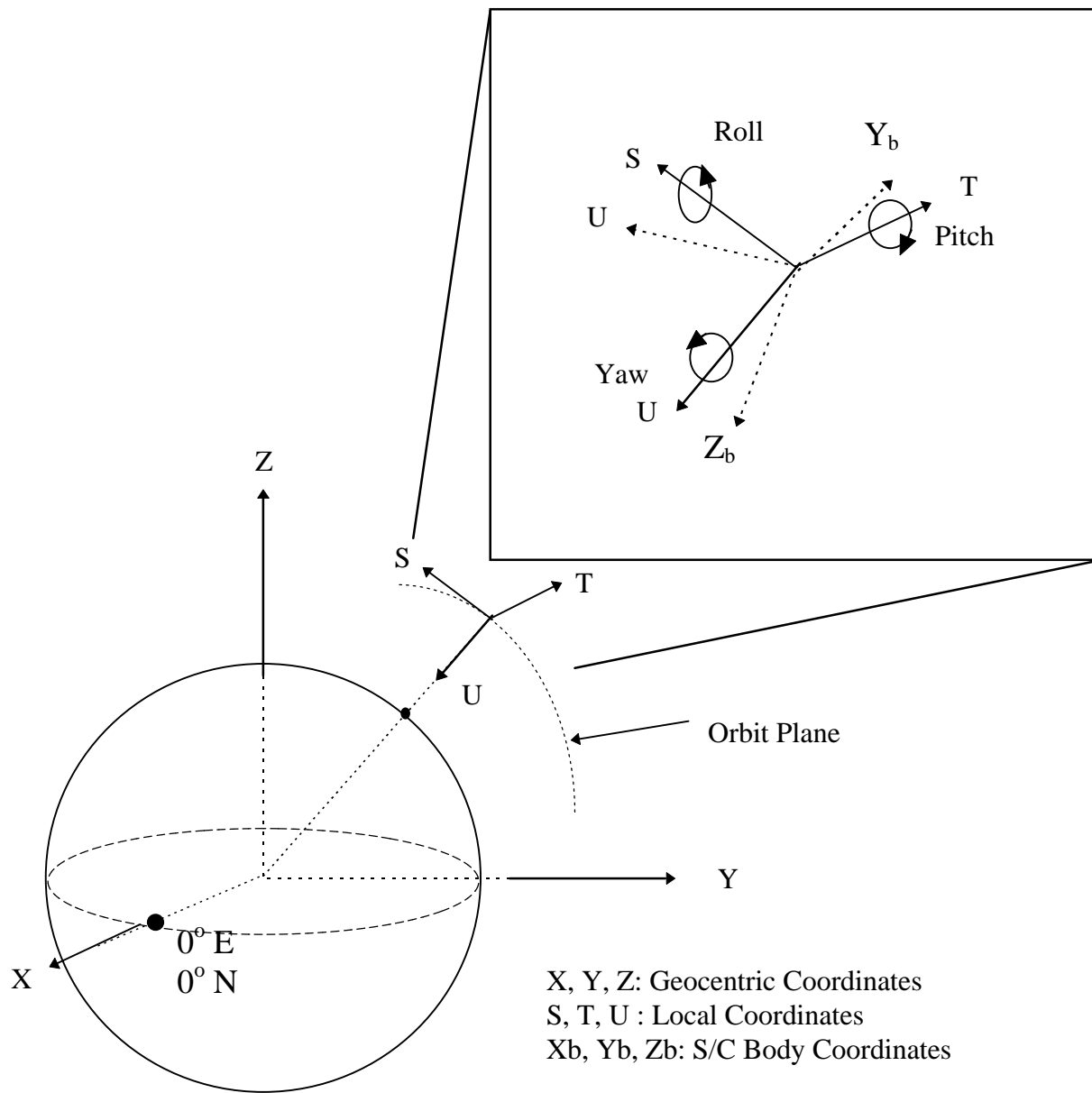


Figure.1 S/C Coordinate Systems

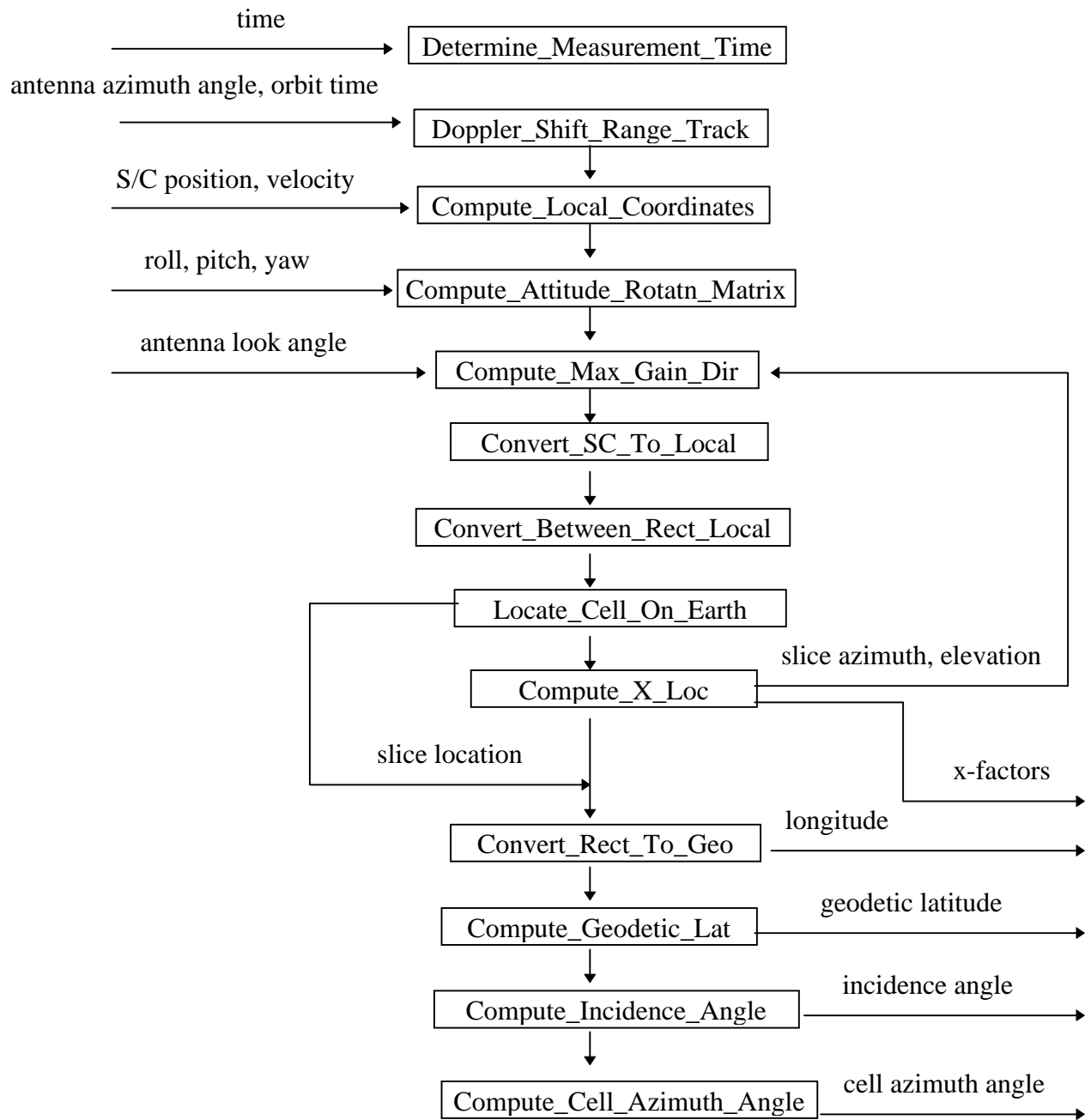


Figure 2 Geometry Module Flow Chart

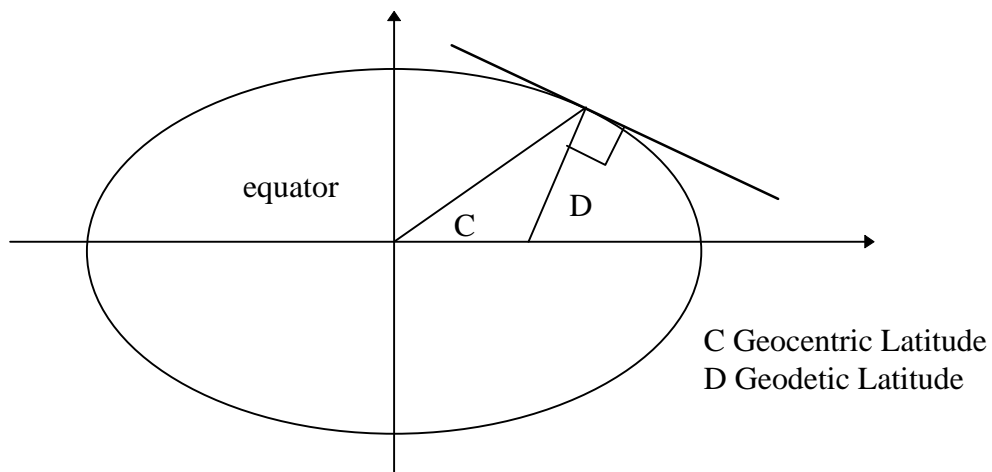


Figure 3. Geodetic Latitude

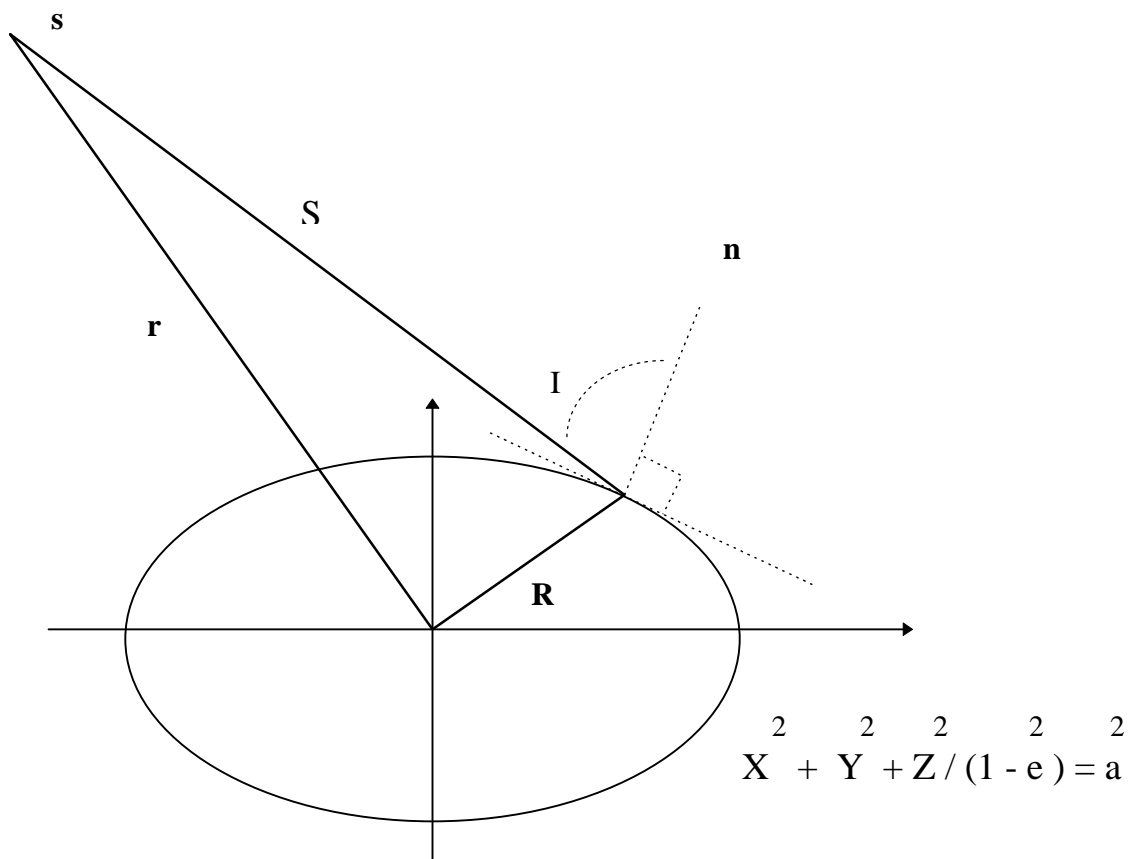


Figure 4. Cell Location

SeaWinds Algorithm Specification

TITLE: LOCAL COORDINATE SYSTEM
SUBMODULE: General Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.2.1
VERSION: 1.1
DATE: 03/31/99
AUTHOR: S. Vincent Hsiao
SUBROUTINE: COMPUTE_LOCAL_COORD
LANGUAGE: FORTRAN
HERITAGE: SEASAT, NSCAT

L1B.2.2.1 Compute_Local_Coord

PURPOSE:

To calculate unit vectors of the axes of the s/c-centered local coordinate system. The outputs are used for the transformation between geocentric and local coordinate systems.

BACKGROUND:

The local rectangular coordinate system is defined by (See Figure 1):

origin - spacecraft (s/c)
axis S - points to $\underline{\mathbf{T}} \times \underline{\mathbf{U}}$
axis T - points to $\underline{\mathbf{U}} \times \underline{\mathbf{v}}$
axis U - points to $-\underline{\mathbf{r}}$

where $\underline{\mathbf{v}}$ is the s/c inertial velocity and $\underline{\mathbf{r}}$ is the s/c position in the geocentric rectangular coordinate system. If the s/c orbit eccentricity were 0, axis S would coincide with $\underline{\mathbf{v}}$.

PROCESSING:

- Step 1. Calculate unit vector $\underline{\mathbf{u}} = -\underline{\mathbf{r}} / r$
Step 2. Calculate unit vector $\underline{\mathbf{t}} = \underline{\mathbf{u}} \times \underline{\mathbf{v}} / v$
Step 3. Calculate unit vector $\underline{\mathbf{s}} = \underline{\mathbf{t}} \times \underline{\mathbf{u}}$

INPUTS:

$\underline{\mathbf{r}}$ position vector of s/c in geocentric coordinate, km
 $\underline{\mathbf{v}}$ inertial velocity of s/c in geocentric coordinate, km/s

OUTPUTS:

s unit vector in S direction
t unit vector in T direction
u unit vector in U direction

REFERENCES:

1. JPL 622-14, SEASAT-A Instrument Data Processing System Detailed Functional Specification, Vol. 1, June 1977
2. JPL D-5610, Science Algorithm Specifications for the NASA Scatterometer Project

SeaWinds Algorithm Specification

TITLE: ATTITUDE MATRIX
SUBMODULE: General Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.2.2
VERSION: 1.1
DATE: 03/31/99
AUTHOR: S. Vincent Hsiao
SUBROUTINE: COMPUTE_ATTITUDE_ROTATN_MATRIX
LANGUAGE: FORTRAN
HERITAGE: NSCAT

L1B.2.2.2 Compute_Attitude_Rotatn_Matrix

PURPOSE:

Construct a matrix for coordinate transformation due to roll, pitch, and yaw of s/c.

BACKGROUND/PROCESSING:

The spacecraft body coordinate system is defined by:

origin - s/c
 axis X_b - to the front of s/c
 axis Y_b - to the right of s/c
 axis Z_b - to the nadir side of s/c.

If roll, pitch, and yaw are all zero, the spacecraft body coordinate system coincides with the local coordinate system. The s/c body coordinate system can be obtained by a T-S-U ordered rotation of axes by pitch(P), roll(R), and yaw(Y) angles of local coordinate system.

A vector in the s/c system can be converted to the local system by [see Ref. 2]:

$$\begin{bmatrix} S \\ T \\ U \end{bmatrix} = \begin{bmatrix} \cos(P) & 0 & \sin(P) \\ 0 & 1 & 0 \\ -\sin(P) & 0 & \cos(P) \end{bmatrix}^{-1} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(R) & -\sin(R) \\ 0 & \sin(R) & \cos(R) \end{bmatrix}^{-1}$$

$$\begin{bmatrix} \cos(Y) & -\sin(Y) & 0 \\ \sin(Y) & \cos(Y) & 0 \\ 0 & 0 & 1 \end{bmatrix}^{-1} \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix} = \mathbf{A} \begin{bmatrix} X_b \\ Y_b \\ Z_b \end{bmatrix}$$

It is assumed here that the s/c attitude is given in the order of pitch-roll-yaw with respect to the T-S-U axes of local coordinate system. If the attitude sequence is given differently the sequence of matrix multiplication has to be changed accordingly.

INPUTS:

R roll angle, deg
P pitch angle, deg
Y yaw angle, deg

OUTPUT:

A_{ij} , $i=1,3$, $j=1,3$ = attitude matrix

REFERENCES:

1. JPL D-5610, Science Algorithm Specifications for the NASA Scatterometer Project
2. Korn, G. A. and T. M. Korn, Mathematical Handbook for Scientists and Engineers, 2nd ed., McGraw-Hill, 1968

SeaWinds Algorithm Specification

TITLE: S/C BODY TO LOCAL COORDINATE TRANSFORMATION
SUBMODULE: General Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.2.3
VERSION: 1.1
DATE: 03/31/99
AUTHOR: S. Vincent Hsiao
SUBROUTINE: CONVERT_SC_TO_LOCAL
LANGUAGE: FORTRAN
HERITAGE: SEASAT,NSCAT

L1B.2.2.3 Convert_SC_To_Local

PURPOSE:

To transform a vector from the spacecraft (s/c) body coordinate system to the local coordinate system by roll, pitch, and yaw of s/c.

BACKGROUND/PROCESSING:

See the Algorithm Specification for "Attitude Matrix" (L1B.2.2.2) for additional details on the coordinate transformation matrix.

The following calculation is done in this subroutine:

$$O(i) = \sum_{j=1,3} [I(j) \times P(i,j)] \quad i=1,3$$

INPUTS:

I vector in s/c body coordinate system
P(i,j), i=1,3, j=1,3 matrix calculated in subroutine
COMPUTE_ATTITUDE_ROTATN_MATRIX

OUTPUTS:

O transformed vector in local coordinate system

REFERENCES:

1. JPL D-5610, Science Algorithm Specifications for the NASA Scatterometer Project

2. Korn, G. A. and T. M. Korn, Mathematical Handbook for Scientists and Engineers, 2nd ed., McGraw-Hill, 1968

SeaWinds Algorithm Specification

TITLE: GEOCENTRIC-LOCAL TRANSFORMATION
SUBMODULE: General Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.2.4
VERSION: 1.1
DATE: 03/31/99
AUTHOR: S. Vincent Hsiao
SUBROUTINE: CONVERT_BETWEEN_RECT_LOCAL
LANGUAGE: FORTRAN
HERITAGE: SEASAT, NSCAT

L1B.2.2.4 Convert_Between_Rect_Local

PURPOSE:

Perform coordinate transformations between the geocentric rectangular and the s/c-centered local coordinate systems.

BACKGROUND/PROCESSING:

If \underline{s} , \underline{t} , and \underline{u} are unit vectors of the S, T, and U axes of the local coordinate system in geocentric coordinates, respectively, then vector \underline{A} in the geocentric coordinate system can be transformed to vector \underline{B} in the local coordinate system by:

$$\begin{aligned} B(1) &= A(1)*s(1) + A(2)*s(2) + A(3)*s(3) \\ B(2) &= A(1)*t(1) + A(2)*t(2) + A(3)*t(3) \\ B(3) &= A(1)*u(1) + A(2)*u(2) + A(3)*u(3). \end{aligned}$$

The inverse transformation is performed by:

$$\begin{aligned} A(1) &= B(1)*s(1) + B(2)*t(1) + B(3)*u(1) \\ A(2) &= B(1)*s(2) + B(2)*t(2) + B(3)*u(2) \\ A(3) &= B(1)*s(3) + B(2)*t(3) + B(3)*u(3). \end{aligned}$$

INPUTS:

\underline{s} unit vector of S axis in geocentric coordinates
 \underline{t} unit vector of T axis in geocentric coordinates
 \underline{u} unit vector of U axis in geocentric coordinates
 \underline{A} if transformation is geocentric --> local

B if transformation is local --> geocentric

OUTPUT:

B if transformation is geocentric --> local

A if transformation is local --> geocentric

REFERENCES:

JPL 622-14, SEASAT-A Instrument Data Processing System Detail Functional Specification,
Vol. 1, June 1977

JPL D-5610, Science Algorithm Specifications for the NASA Scatterometer Project

SeaWinds Algorithm Specification

TITLE: RECTANGULAR SYSTEM TO LONG-LAT TRANSFORMATION
SUBMODULE: General Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.2.5
VERSION: 1.1
DATE: 03/31/99
AUTHOR: S. Vincent Hsiao
SUBROUTINE: CONVERT_RECT_TO_GEO
LANGUAGE: FORTRAN
HERITAGE: NSCAT

L1B.2.2.5 Convert_Rect_To_Geo

PURPOSE:

Calculate longitude and geocentric latitude from a position vector.

BACKGROUND/PROCESSING:

The following equations transform a position vector (x,y,z) in the geocentric rectangular coordinate system to longitude (G) and geocentric latitude (T):

$$H = \text{SQRT}(x*x + y*y + z*z)$$

$$G = \text{ARCTAN2}(y,x)$$

$$T = \text{ARCSIN}(z/H).$$

INPUTS:

(x,y,z) position vector of the center of an instantaneous sigma-0 cell, km

OUTPUTS:

G longitude of the center or a corner of an instantaneous sigma-0 cell, deg.

T geocentric latitude of the center or a corner of an instantaneous sigma-0 cell, deg.

REFERENCE:

1. JPL D-5610, Science Algorithm Specifications for the NASA Scatterometer Project

SeaWinds Algorithm Specification

TITLE: GEOCENTRIC TO GEODETIC LATITUDE TRANSFORMATION
SUBMODULE: General Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.2.6
VERSION: 1.1
DATE: 03/31/99
AUTHOR: S. Vincent Hsiao
SUBROUTINE: COMPUTE_GEODETIC_LAT
LANGUAGE: FORTRAN
HERITAGE: NSCAT

L1B.2.2.6 Compute_Geodetic_Lat

PURPOSE:

To calculate geodetic latitude from geocentric latitude.

BACKGROUND/PROCESSING:

The geodetic latitude (D) is defined in Figure 3. It is the latitude on regular maps. However, the geocentric latitude (C) is used in most of the geometry computations because it is easier to calculate. To convert from geocentric to geodetic latitude the following equation can be used (See Reference 1):

$$D = \tan^{-1} [\tan C / (1-e^2)].$$

where e is the ellipticity of the Earth.

INPUTS:

C geocentric latitude, deg

OUTPUTS:

D geodetic latitude, deg

REFERENCES:

1. Escobal, Theory of Orbit Determination, 1968.
2. JPL D-5610, Science Algorithm Specifications for the NASA Scatterometer Project.

SeaWinds Algorithm Specification

TITLE: BORESIGHT
SUBMODULE: Antenna Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.3.1
VERSION: 1.1
DATE: 03/31/99
AUTHORS: S.V. Hsiao
SUBROUTINE: COMPUTE_MAX_GAIN_DIR
LANGUAGE: FORTRAN
HERITAGE: SeaWinds ATB Compute_Max_Gain_Dir

L1B.2.3.1 Compute_Max_Gain_Dir

PURPOSE:

Calculate the antenna maximum gain direction in the s/c body coordinate system.

BACKGROUND/PROCESSING:

The maximum gain direction of an antenna in the s/c body coordinate system is obtained by rotating (0,0,1) by antenna azimuth(A) and look angle(L) with respect to the 3rd and 2nd axes of the s/c body coordinate system, i.e.

$$F(1) = \text{SIN}(L) * \text{COS}(A)$$

$$F(2) = \text{SIN}(L) * \text{SIN}(A)$$

$$F(3) = \text{COS}(L)$$

INPUTS:

L antenna look angle, deg
A antenna azimuth angle, deg

OUTPUTS:

F unit vector of the antenna maximum gain direction in s/c body coordinate system

SeaWinds Algorithm Specification

TITLE: CALCULATION OF MEASUREMENT TIME
SUBMODULE: Antenna Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.3.3
VERSION: 1.1
DATE: 03/31/99
AUTHORS: S.V. Hsiao and A. Zhang
SUBROUTINE: Determine_Measurement_Time
LANGUAGE: FORTRAN
HERITAGE: SeaWinds ATB

L1B.2.3.3 Determine_Measurement_Time

PURPOSE

Determine the measurement time for every pulse. This time is defined as the time the antenna beam strikes the Earth's surface, which, generally speaking, is half way between the transmission time and the receiving time. The adjustment in this algorithm is based on the first transmitting timetag value of the data frame.

REQUIREMENT

The error for a determined timetag value at the receiving time should be less than 1.0×10^{-4} sec.

BACKGROUND

The timetag of the first transmission time is reported by every telemetry data package. The pulse time for every pulse needs to be determined in order to correctly calculate the location of the beam on the Earth's surface. This algorithm determines the pulse time by two steps:

- (1) determination of a transmitting time for a pulse, using the frame time, and
- (2) adjustment of the timetag from a transmission time to a pulse time.

For a specific antenna beam, the difference in slant range between the two successive pulses is less than 8km for outer beam and 5.5 km for inner beam, respectively. The corresponding time errors are within our error budgets.

Based on this fact, the first step is to determine the time difference between the frame timetag and a beam's transmitting time:

$$\text{delt1} = \text{float}(\text{num_in_frame}-1) / \text{prf}. \quad (1)$$

Then a transmitting timetag is determined by adding delt1 to the frame time.

$$\text{timetag} = \text{frame_timetag} + \text{delt1} \quad (2)$$

Finally, the timetag is adjusted to match the time that the beam hits the Earth using the slant range of the previous pulse ($\text{r_slant_old}(\text{ibeam})$) of the same antenna beam:

$$\text{delt2} = \text{r_slant_old}(\text{ibeam})/C. \quad (3)$$

$$\text{timetag} = \text{timetag} + \text{delt2} \quad (4)$$

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Dimension</u> | <u>Description</u> |
|----------------------|--------------|------------------|---|
| frame_timetag | sec | | timetag at transmitting time of the first pulse |
| num_in_frame | | | the current pulse number in the data frame |
| prf | Hz | | pulse repetition frequency |
| r_slant_old | Km | | slant range of previous pulse |
| ibeam | | | beam ID (inner/outer) |

OUTPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Dimension</u> | <u>Description</u> |
|----------------------|--------------|------------------|------------------------------------|
| timetag | sec | | timetag for the current pulse time |
| delt2 | sec | | time adjustment |

PROCESSING

1. Calculate the time difference between the frame time and the current transmitting time by using eq. (1).
2. Determine the current transmitting time by eq.(2).
3. Calculate the time adjustment by using eq. (3). If it is the first pulse of processing, or it is the first pulse after a data gap, then use the nominal slant range instead of the previous range in eq.(3).
4. Calculate the current pulse time by eq. (4).

CONSTANTS

| | |
|----------------|--|
| C | the speed of light |
| r_slant_nom(2) | nominal slant range for inner / outer beam |

INTERNAL VARIABLES

| <u>Variable Name</u> | <u>Units</u> | <u>Dimension</u> | <u>Description</u> |
|----------------------|--------------|------------------|--|
| delt1 | sec | | time difference between current transmission time and frame time |

SeaWinds Algorithm Specification

TITLE: LOCATE CELL CENTER
SUBMODULE: Cell Location and Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.4.1
VERSION: 1.1
DATE: 03/31/99
AUTHOR: S. Vincent Hsiao
SUBROUTINE: LOCATE_CELL_ON_EARTH
LANGUAGE: FORTRAN
HERITAGE: SEASAT, NSCAT

L1B.2.4.1 Locate_Cell_On_Earth

PURPOSE:

Compute the intersection of the vector from the spacecraft to the cell center with the Earth's surface.

BACKGROUND:

The Earth is assumed to be an oblate ellipsoid here. The equations to be solved for the cell center position are [see Figure 4]:

$$X^{**2} + Y^{**2} + Z^{**2}/(1-e*e) = a^{**2} \quad [1]$$

$$\underline{\mathbf{r}} + S * \underline{\mathbf{s}} = \underline{\mathbf{R}} \quad [2]$$

where:

$\underline{\mathbf{R}}$ = (X,Y,Z) is the position of cell center,
 e = eccentricity of the Earth,
 a = semi-major axis of the Earth,
 $\underline{\mathbf{r}}$ = (x,y,z) is the position vector of s/c in geocentric system,
 S = slant range,
 $\underline{\mathbf{s}}$ = unit vector in the direction from s/c to the cell center in geocentric system.

By substituting X, Y, and Z from Equation [2], Equations [1] can be reduced to

$$C1*S^{**2} + 2*C2*S + C3 = 0 \quad [3]$$

where

$$C1 = s(1)**2 + s(2)**2 + s(3)**2/(1-e**2) \quad [4]$$

$$C2 = x*s(1) + y*s(2) + z*s(3)/(1-e**2) \quad [5]$$

$$C3 = x**2 + y**2 + z**2/(1-e**2) - a**2. \quad [6]$$

PROCESSING:

Step1. Calculate C1, C2, and C3 using [4], [5], and [6].

Step2. Solve the quadratic equation [3] for S.

Step3. Calculate **R** using [2].

If equation [3] has two real roots, the larger one, which represents a point on the far side of the Earth's surface, is discarded. Complex roots indicate that there is no solution on the Earth's surface. In this case a flag (If) is set and the next cell is processed.

INPUTS:

r s/c position vector, km

s unit vector from s/c to the cell center

OUTPUTS:

R cell center position vector, km

If flag indicating whether beam hit the Earth's surface, 1=yes, -1=no

AUXILIARY DATA:

a semi-major axis of the Earth, km

e eccentricity of the Earth

REFERENCES:

[1] JPL 622-14, SEASAT-A Instrument Data Processing System Detail Functional Specification, Vol. 1, June 1977

[2] JPL D-5610, Science Algorithm Specifications for the NASA Scatterometer Project.

SeaWinds Algorithm Specification

TITLE: INCIDENCE ANGLE
SUBMODULE: Cell Location and Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.4.2
VERSION: 1.1
DATE: 03/31/99
AUTHOR: S. Vincent Hsiao
SUBROUTINE: COMPUTE_INCIDENCE_ANGLE
LANGUAGE: FORTRAN
HERITAGE: SEASAT, NSCAT

L1B.2.4.2 Compute_Incidence_Angle

PURPOSE:

Calculate the incidence angle at cell center.

BACKGROUND:

The incidence angle is defined as the angle between the local normal vector at the Earth's surface and the s/c to cell center direction vector [see Figure 4]. The local normal at cell center, $\underline{\mathbf{R}} = (X, Y, Z)$, is

$$\underline{\mathbf{n}} = (2*X/a^{**2}, 2*Y/a^{**2}, 2*Z/b^{**2}) \quad [1]$$

where a is the semi-major axis of the Earth and b is the semi-minor axis of the Earth. The incidence angle can be calculated by

$$I = 180 - \text{ARCCOS} [\text{DOT}(\underline{\mathbf{s}}, \underline{\mathbf{n}}) / n] \quad [2]$$

where $\underline{\mathbf{s}}$ is the unit vector of $(\underline{\mathbf{R}} - \underline{\mathbf{r}})$ and $\underline{\mathbf{r}}$ is the position vector of the s/c.

PROCESSING:

Step1. Calculate $\underline{\mathbf{n}}$ using [1];

Step2. Calculate I using [2].

INPUTS:

$\underline{\mathbf{R}}$ position vector of cell center, km

OUTPUTS:

I incidence angle, deg

AUXILIARY DATA:

a semi-major axis of the Earth, km

b semi-minor axis of the Earth, km

INTERNAL DATA:

$\underline{\mathbf{s}}$ unit vector of ($\underline{\mathbf{R}} - \underline{\mathbf{r}}$), from slant range calculation.

REFERENCES:

[1] JPL 622-14, SEASAT-A Instrument Data Processing System Detailed Functional Specification, Vol. 1, June 1977

[2] JPL D-5610, Science Algorithm Specifications for the NASA Scatterometer Project

SeaWinds Algorithm Specification

TITLE: CELL AZIMUTH ANGLE
SUBMODULE: Cell Location and Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.4.4
VERSION: 1.1
DATE: 03/31/99
AUTHOR: S. Vincent Hsiao, A. Zhang
SUBROUTINE: COMPUTE_CELL_AZIMUTH_ANGLE
LANGUAGE: FORTRAN
HERITAGE: SeaWinds ATB

L1B.2.4.4 Compute_Cell_Azimuth_Angle

PURPOSE:

Calculate the cell azimuth angle from the north.

BACKGROUND:

The cell azimuth angle is the clockwise angle measured from the north to the s/c to cell center direction vector \underline{s} [see Figure 4] then projected on the north-east plane in the geographic coordinates (see L1B.2.2.7).

From the geocentric latitude and longitude of the cell center, a transformation matrix between the geocentric and geographic coordinates is computed using L1B.2.2.7. The vector \underline{s} is then transformed to the geographic coordinates. The azimuth angle is computed by $ATAN2(s_E, s_N)$ where s_E , and s_N are the east and north components of \underline{s} , respectively.

PROCESSING:

Step1. Call L1B.2.2.5 to calculate s/c longitude and geocentric latitude;

Step2. Call L1B.2.2.7 to calculate transformation matrix;

Step3. Convert \underline{s} to the geographic coordinates;

Step4. Compute azimuth angle by $ATAN2(s_E, s_N)$.

INPUTS:

\underline{s} = s/c to cell center direction vector
s/c position vector

OUTPUTS:

cell azimuth angle

INTERNAL DATA:

transformation matrix from geocentric to geographic coordinate systems

REFERENCE:

[1] Chong-Yung Chi, "Orbital and Geometric Calculations for NSCAT", JPL MEMO 3343-86-272, Dec. 2, 1986

SeaWinds Algorithm Specification

TITLE: DOPPLER_SHIFT_RANGE_TRACKING
SUBMODULE: General Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.4.5
VERSION: 1.1
DATE: 03/31/99
AUTHOR: S. Vincent Hsiao
SUBROUTINE: DOPPLER_SHIFT_RANGE_TRACK
LANGUAGE: FORTRAN
HERITAGE: None

L1B.2.4.5 Doppler_Shift_Range_Track

PURPOSE:

Duplicate the CDS computation of Doppler shift and range tracking in order to compute the base-band frequency accurately.

BACKGROUND:

The Doppler shift , f_d , in Hz is computed by:

$$f_d = a_0 + a_1 \cos(\alpha + p_1) \quad [1]$$

where

$$a_0 = a_{0m} C + a_{0b} \quad [2]$$

$$a_1 = a_{1m} A + a_{1b} \quad [3]$$

$$p_1 = p_{1m} P + p_{1b} , \quad [4]$$

α is the true antenna azimuth angle and C, A, and P values are extracted from the Doppler shift table indexed in orbit steps. The range delay, D_r , in ms is computed the same way except using the range delay table and a different set of a_{0m} , a_{0b} , a_{1m} , a_{1b} , p_{1m} , and p_{1b} values. In Eq. [1], in order to match the CDS computation, a cosine table is used instead of the cosine function.

The true antenna azimuth angle α in DN is computed by:

$$\alpha_{DN} = \alpha_r + \delta_b + \delta_c + \delta_e \quad [5]$$

where α_r is the raw azimuth angle, δ_b is the beam offset, δ_c is the centering offset, δ_e is the

encoder offset, and

$$\delta_i = \text{pri} * \text{cds_spin_rate} \quad [6]$$

is the internal offset. The centering offset, δ_c , is computed by:

$$\delta_c = \delta_i (\text{range_delay_prev_pulse} + \text{pulse_width}) / 2 / \text{pri} . \quad [7]$$

Note that when the above computation is executed, the `range_delay_prev_pulse` and `pulse_width` are in DN's with scale of 0.049903 ms/DN but the `pri` is in DN with scale of 0.099806ms/DN. Thus the actual computation is

$$\delta_c = \delta_i (\text{range_delay_prev_pulse}_{\text{DN}} + \text{pulse_width}_{\text{DN}}) / 4 / \text{pri}_{\text{DN}} . \quad [8]$$

The quantization of azimuth angles is such that 32768 DN equals one full circle.

The commanded delay, D_c , in real DN is

$$D_c = D_r / R_D - (\text{range_gate_width}_{\text{DN}} - \text{pulse_width}_{\text{DN}}) / 2. \quad [9]$$

where R_D ($= 0.049903$ ms/DN) is the delay resolution.

The commanded Doppler frequency, f_c , in Hz is

$$f_c = -R_f \times \text{nint} (\{ f_d + \mu [D_c - \text{nint}(D_c)] \} / R_f) \quad [10]$$

where R_f ($= 2000$ Hz/DN) is the frequency resolution and μ is the chirp rate.

The commanded delay, D_c , in ms is

$$D_c = \text{nint}(D_c) R_D . \quad [11]$$

The base-band frequency, f_{bb} , which will be used in computing X factors and slice locations, is:

$$f_{bb} = (-f_c + \mu D_c) - (f_p + \mu D_p) + \mu \times D_s \quad [12]$$

where

$$D_s = (\text{range_gate_width} - \text{pulse_width}) / 2. - C_d \quad [13]$$

is the frequency delay offset, C_d is a constant, and $D_p = 2(\text{slant_range})/\text{speed_of_light}$.

The cosine table has 256 entries corresponding to argument 0 to $2\pi \times 255/256$. The cosine values are linearly interpolated.

PROCESSING:

1. Compute the internal offset (Eq. [6]) and round it to integer.
2. Compute the true antenna azimuth angle in real DN (Eqs. [5] and [8]).
3. Convert the true antenna azimuth angle from DN to radians.
4. Compute a_0 , a_1 , and p_1 for range delay (Eqs. [2], [3], and [4]).
5. Compute the range delay in ms (Eq. [1]).
6. Convert the range delay to real DN, save the value to `range_delay_prev_pulse`, and compute the commanded range delay in real DN (Eq. [9]).
7. Convert the commanded range delay to (rounded) integer DN.
8. Compute a_0 , a_1 , and p_1 for Doppler shift (Eqs. [2], [3], and [4]).
9. Compute Doppler shift in Hz (Eq. [1]).
10. Compute the commanded Doppler shift in Hz (Eq. [10]).
11. Convert the commanded range delay from DN to ms (Eq. [11]).
12. Compute the D_s in ms (Eq. [13]).

INPUTS:

1. Doppler step, an integer ranging from 0 to 255.
2. Raw antenna azimuth angle in DN from L1A.
3. Beam number, 1=inner, 2=outer.
4. Encoder number.
5. Range gate width in DN from L1A.
6. Pulse width in DN from L1A.
7. `pri` (pulse repetition interval) in DN from L1A.

OUTPUTS:

1. Commanded Doppler shift in Hz.
2. Commanded range delay in ms.
3. Delay shift in ms.

AUXILIARY DATA:

1. $\text{cds_spin_rate (in DN/ms)} = \text{spin_rate(rpm)} / 60 \times 32768 / 1000$.
2. Chirp rate (μ) in kHz/ms.
3. Constant ($C_D = 0.04905722$ ms) in computing delay_shift. It is computed from three constants given in Ref. 3.
4. Beam offset, two values for inner and outer beams.
5. Encoder offset, two values for two encoder flags.
6. Cosine table, array of 256 real numbers.

REFERENCE:

- [1] Glenister, R., SeaWinds Doppler and Range Algorithms, JPL Interoffice Memorandum 3347-98-002, Jan. 15, 1998
- [2] Glenister, R., The CDS Implementation of the Doppler Shift and Range Tracking Algorithms, JPL Interoffice Memorandum 3347-98-051, Sep. 15, 1998
- [3] Glenister, R., Frequency and Gate Centering Scheme, JPL Interoffice Memorandum 3347-98-057, Oct. 9, 1998

SeaWinds Algorithm Specification

TITLE: COMPUTE_X_LOC
SUBMODULE: General Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.4.6
VERSION: 1.1
DATE: 07/31/01
AUTHOR: S. Vincent Hsiao
SUBROUTINE:
LANGUAGE: FORTRAN
HERITAGE: None

L1B.2.4.6 Compute_X_Loc

PURPOSE:

To compute the X factor values and slice locations from the tables.

BACKGROUND:

Using the table, the X factor is computed by

$$X = X_{\text{nom}} + B\Delta f + C\Delta f^2 + D\Delta f^3 \quad [1]$$

where X_{nom} , B, C, and D are interpolated table values, and

$$\Delta f = f_{\text{bb}} / \delta f + SH \quad [2]$$

where S is interpolated from the table, H is the local elevation extracted from the elevation map of _ degree resolution, δf is the width of FFT bins, and f_{bb} is the base-band frequency shift computed as.

$$f_{\text{bb}} = \mu (D_c + D_s - 2R/c) - f_c - f_p \quad [3]$$

where μ is the chirp rate, D_c is the commanded range delay, D_s is a delay offset term computed in Algorithm L1B.2.4.5 (DOPPLER_SHIFT_RANGE_TRACKING), R is the slant range at the boresight, c is the speed of light,) f_c is the commanded Doppler frequency, and f_p is the Doppler frequency at the boresight.

The slice locations are computed by

$$\alpha_{\text{az}} = A_{\text{az}} + B_{\text{az}} \Delta f \quad [4]$$

$$\alpha_{el} = A_{el} + B_{el} \Delta f \quad [5]$$

where α_{az} and α_{el} are azimuth and elevation angle offsets from the boresight, A_{az} , B_{az} , A_{el} and B_{el} are interpolated values from the table.

Using the tables, the maximum errors for the inner beam are 0.12 dB for X, 0.009 dB for X egg, 0.024 degrees for the azimuth angle, and 0.008 for the elevation angle. For the outer beam the maximum errors are 0.05 dB for X, 0.012 dB for X egg, 0.018 degrees for the azimuth angle, and 0.0045 degrees for the elevation angle. See Reference 3.

PROCESSING:

1. Compute the orbit time index by $[(32.*orbit_time/orbit_period) \bmod 32]$.
2. Compute the antenna azimuth angle index by $[36.*antenna_azimuth_angle/360]$.
3. Using bilinear interpolation, compute the S factor; A_{az} , B_{az} , A_{el} and B_{el} for 8 center slices; and X_{nom} , B, C, and D for 8 center slices and for egg.
4. Compute f_{bb} Using Eq. [3].
5. Compute H:
 $i = \text{mod}(\text{nint}(\text{longitude}/0.25), 1440) + 1$
 $j = \text{nint}((\text{latitude} + 90.)/0.25) + 1$
 $H = \text{Height}(i, j)$
6. Compute Δf using Eq. [2].
7. Compute X using Eq. [1].
8. Compute α_{az} and α_{el} Using Eqs. [4] and [5].

INPUTS:

1. Beam number, an integer, 1=inner beam, 2=outer beam.
2. Resolution mode, an integer, valid value ranges from 1 to 8, however, 8 is not used.
3. Orbit time in seconds.
4. Antenna azimuth angle in degrees.
5. Latitude of boresight ground point, in degrees.

6. Longitude of boresight ground point, in degrees.
7. Doppler frequency at boresight in Hz.
8. Commanded Doppler frequency in Hz.
9. Slant range at boresight in meters.
10. Commanded range delay in seconds
11. Delay offset in seconds.

OUTPUTS:

1. X factors for eight center slices and egg.
2. Azimuth and elevation angle offsets from the boresight for eight center slices.

AUXILIARY DATA:

1. Chirp rate in Hz/s.
2. Light speed in m/s.
3. X_{nom} , B, C, and D tables, real arrays with dimension (9,2,36,32,8), the first index indicates the eight slices and egg, the second index is the beam number, the third is the antenna azimuth angle index, the fourth is the orbit time index, and the fifth is the resolution mode.
4. S table, real array with dimension (2,36,32,8), the indices are the same as the last four indices of the above.
5. A_{az} , B_{az} , A_{el} and B_{el} tables, real arrays with dimension (8,2,36,32,8); the indices are the same as item 3 except for the first index, here the values for eggs are not needed.
6. Height data in meters, a two-byte integer array of dimension (1440,721), each element Height(i,j) in the array indicates the elevation of a 0.25×0.25 degree area centered at $(0.25*(i-1))$ east longitude and $(0.25*(j-1)-90.)$ latitude.
7. Orbit period in seconds.
8. FFT bin width in Hz.

REFERENCES:

1. X Factor Utilization Algorithms, Spencer, M. et.al., September 28,1998, Jet Propulsion Laboratory, Interoffice Memorandum 3347-98-54.
2. Correcting X for Topography, Ashcraft, I.S., December 17, 1998, Brigham Young University, MERS Technical Report # MERS 98-07
3. Background and Accuracy Analysis of the Xfactor7 Table: Final Report on SeaWinds X Factor Accuracy, Jones, B.E., et. al., January 13,1999, Brigham Young University, MERS Technical Report # MERS 99-01.
- 4 . How to Produce and Use the X-factor Table, Ashcraft, I.S., January 12,1999, Brigham Young University, MERS Technical Report # MERS 99-03.

SeaWinds Algorithm Specification

TITLE: SELECT BEST EIGHT SLICES
SUBMODULE: Cell Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.4.7
VERSION: 1.0
DATE: 05/15/01
AUTHOR: S. Vincent Hsiao
SUBROUTINE:
LANGUAGE: FORTRAN90
HERITAGE: None

L1B.2.4.7 Select_Best_Eight_Slices

PURPOSE:

To select the best eight consecutive slices out of the center ten slices of an echo spectrum.

BACKGROUND:

The distribution of echo power in the ten center slices of a measurement pulse is affected by the orbit and attitude of the spacecraft. In the data processing, eight consecutive slices, out of the center ten, that best represent the measurement are used. The outer two slices (1st and 12th slices) are "guard" slices and are not used. The selection of the best eight (center eight or "higher" or "lower" eight) can be decided by the "frequency shift" value which contains the effects of orbit and attitude.

The existing QuikSCAT data have been examined, confirming that the computed "frequency shift" is a good indicator of where the power peak is.

INPUTS:

1. frequency shift in Hz
2. resolution mode of the measurement (1 to 8)

OUTPUTS:

starting index of the best eight slices, possible values are 2, 3, and 4.

AUXILIARY DATA:

echo band width of the center ten data slices in Hz, real(8) array

PROCESSING:

Step 1. Set the starting index to 3.

Step 2. Compute $(\text{frequency_shift})/[\text{10_slice_echo_band_width}(\text{resolution_mode})]/10$.

Step 3. If the result in Step 2. is less than -0.5, set the starting index to 2.

Step 4. If the result in Step 2. is greater than 0.5, set the starting index to 4.

SeaWinds Algorithm Specification

TITLE: CALCULATE FREQUENCY LINE
SUBMODULE: Cell Location and Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.4.8
VERSION: 1.0
DATE: 08/31/01
AUTHOR: S. Vincent Hsiao
SUBROUTINE: Locate_Freq_Lines.F
LANGUAGE: FORTRAN
HERITAGE: QuikSCAT

L1B.2.4.8 Locate_Freq_Lines

PURPOSE:

Calculate contour lines of base-band frequencies corresponding to the center of slices.

BACKGROUND:

For each pulse, the return power is further separated into “slices” using FFT. The baseband frequency of a target on the ground is dependent on Doppler frequency and slant range:

$$f = 2\mu(s - s_c)/c - (f_d - f_{d,c}) \quad [1]$$

where μ is the chirp rate, c is the speed of light, s is the slant range, s_c is the commanded slant range, f_d is the Doppler frequency, and $f_{d,c}$ is the commanded Doppler frequency.

In this algorithm, the frequency contour lines corresponding to the center of slices near the antenna bore sight is computed.

The frequency in Hz at the center of slices, f_s , can be computed as:

$$f_s = (n_s - 5.5)\Delta f - 230.9 \quad [2]$$

where Δf is slice width in Hz, n_s is slice number from 1 to 10. The constant 230.9 in Equation [1] is there because at the boundary between slice five and six the frequency is negative of one half of the FFT bin width (461.8 Hz).

It is assumed that the frequency contours are locally straight lines in the azimuth_angle-look_angle surface. The task of locating a line becomes finding two points on the line. This is

done by iteratively finding two look angles on the contour line with corresponding azimuth angles of ± 0.9 degree from pulse center.

INPUTS:

1. commanded Doppler frequency, computed in Calculate_Range_Doppler
2. commanded range, computed in Calculate_Range_Doppler
3. slice width in Hz, resolution mode dependent
4. beam index, indicate inner or outer beam
5. roll-pitch-yaw matrix for coordinate transformation due to attitude, computed in Calculate_Attitude_Rotation_Matrix
6. velocity of s/c in local coordinate system(for Doppler frequency computation), computed in Calculate_Relative_Velocity
7. rotation matrix between local and geocentric coordinate systems, computed in Calculate_Local_Coord
8. s/c velocity from ephemeris
9. s/c position from ephemeris
10. antenna azimuth angle
11. antenna look angle

OUTPUTS:

relative_look_angle

the relative look angles (to the pulse center look angle) at the frequency contours corresponding to the 8 slice centers and the azimuth angles of ± 0.9 degree from the pulse center, real array of dimension (2,8), the first index indicates -0.9 and $+0.9$ degree relative azimuth angles and the second index indicates the 8 slices being used

PROCESSING:

Set azimuth_angle = antenna_azimuth $- 0.9$. Set look_angle_trial = antenna_look $- 0.8$ as initial value for iteration, then loop through each slice of data in the order of increasing look angle (reverse slice order):

1. set $df_+ = 0$. and $df_- = 0$.
2. compute the base-band frequency of the slice, f_s , by Eq. [2].
3. using azimuth_angle and look_angle_trial, by calling L1B.2.3.1 (Calculate_Boresight), L1B.2.2.7 (Calculate_Relative_Velocity), and L1B.2.2.3 (Convert_SC_To_Local) to compute the slant range and Doppler frequency.
4. compute the frequency, f , by Eq. [1], and compute $df = f - f_s$
5. if $|df|$ is less than 10 Hz, the look angle is found, then compute the relative look angle (look_angle_trial - antenna_look), put into the output array, and go to step 1 to do the next slice.
6. if not, then compute the next look angle to try by:
 - 6.a. if $df > 0$ then $df_+ = df$ and set $A_+ = \text{look_angle_trial}$

- 6.b. if $df < 0$ then $df_- = df$ and set $A_- = \text{look_angle_trial}$
- 6.c. if $|df_+ \bullet df_-| > 0.01$, the solution is bracketed, then compute the next look angle by interpolation: $\text{look_angle_trial} = A_- + (A_+ - A_-)(-df_-)/(df_+ - df_-)$
- 6.d. if not , compute the next look angle by $\text{look_angle_trial} = \text{look_angle_trial} - (df/|df|) \bullet 0.2$
- 7. go to step 3.

When the slice loop is done, set $\text{azimuth_angle} = \text{antenna_azimuth} + 0.9$ and $\text{look_angle_trial} = \text{antenna_look} - 0.8$ and loop through each slice again.

SeaWinds Algorithm Specification

TITLE: LOCATE PEAK GAIN
SUBMODULE: Cell Location and Geometry
MODULE: SeaWinds Geometry
CODE: L1B.2.4.9
VERSION: 1.0
DATE: 08/31/01
AUTHOR: S. Vincent Hsiao
SUBROUTINE: Locate_Peak_Gain.F
LANGUAGE: FORTRAN
HERITAGE: QSCAT

L1B.2.4.9 Locate_Peak_Gain

PURPOSE:

Find the peak gain along the contour lines of base-band frequencies corresponding to the center of a slice.

BACKGROUND:

The location of the center of a slice is represented by the location of peak gain along the contour line corresponding to the center frequency of the slice. In this algorithm, for each slice, the two-way antenna gain along the contour line (computed in L1B.2.4.7) are computed using bi-linear interpolation at azimuth angles from -1.0 to $+1.0$ degrees with 0.05 degree increment. Then a quadratic fit through the highest and two neighboring points is used to find the maximum gain value and its location.

INPUTS:

1. relative_look_angle
relative look angles (to the pulse center look angle) at the frequency contours corresponding to the 8 slice centers and the azimuth angles of ± 0.9 degree from pulse center, real array of dimension (2,8), the first index indicates -0.9 and $+0.9$ degree relative azimuth angles and the second index indicates the 8 slices being used.
2. antenna_gain_in and antenna_gain_out
two-way antenna gain for inner and out beam, respectively, dimension is (-60:60, -150:150), the first index multiplied by 0.05 is the corresponding relative azimuth angle and the second index multiplied by 0.03095356 is the corresponding relative look angle

OUTPUTS:

1. peak_gain
maximum gain along the frequency contour of the cell center frequency, in dB; real array of dimension 8.
2. peak_gain_relative_look_angle
look angle relative to the boresight at the peak gain, in degrees; real array of dimension 8
3. peak_gain_relative_azimuth_angle
azimuth angle relative to the bore sight at the peak gain, in degrees; real array of dimension 8

PROCESSING:

Do for each slice:

1. Compute the look angle increment corresponding to 0.05 degree azimuth angle increment:
 $dl = (\text{relative_look_angle}(2,i) - \text{relative_look_angle}(1,i)) / 36$, where i is the slice number.
2. Compute the starting relative look angle: $\text{angle_look} = \text{relative_look_angle}(1,i) - 2 \bullet dl$
3. Starting with relative look angle of angle_look and relative azimuth angle of -1.0 , with increments of dl and 0.05 , respectively, ending when azimuth angle is greater than $+1.0$, compute the antenna gain, $ag(i)$, $i=1,41$, using bi-linear interpolation in antenna_gain_in or antenna_gain_out array depending on the beam index
4. Find the maximum gain, if the maximum gain is greater than the threshold (-40 dB), and fit the maximum and two neighboring gains through a quadratic equation
5. Compute gain, look angle, and azimuth angle corresponding to the maximum of the quadratic equation

Steps 4 and 5 can be accomplished by:

```
c=ag(j-1) ! j-1 is the index of the maximum ag
b=0.5*(ag(j)-ag(j-2))
a=ag(j)-b-c
xm=-0.5*b/a
peak_gain=xm*(a*xm+b)+c
peak_gain_relative_look_angle=angle_look(j-1)+xm*dl
peak_gain_relative_azimuth_angle=-1.05+0.05*(xm+j-1)
```

QuikSCAT/SeaWinds Sigma0 and Kp Algorithm

MODULE L1B.3.0

ALGORITHM SPECIFICATIONS

**R. Scott Dunbar
Vincent Hsiao
Young-Joon Kim
Kyung Pak
Angela Zhang**

VERSION: 2.0

DATE: October 31, 1999

SeaWinds Sigma0 and Kp Algorithm

I. MODULE OVERVIEW

Introduction

This algorithm specification describes the computation of the radar backscattering coefficient, σ_0 . This computation is performed for each σ_0 measurement of Level 1A data. The output from this module to the Level 1B (L1B) record includes the σ_0 , the conversion factor which relates received signal energy to σ_0 (X_{factor}), the calibration part of the X_{factor} , X_{cal} , and the ratio of signal to noise energy (SNR), as well as other instrument parameters. This section gives a general overview of the algorithms. The individual submodule description in section III provide more detailed background information on σ_0 and Kp processing algorithms.

QuikSCAT/SeaWinds Instrument and Power Detection

The QuikSCAT/SeaWinds instrument is an active microwave radar designed to measure electromagnetic wave scattering from wind roughened ocean surfaces. Unlike its predecessor NASA Scatterometer (NSCAT), a fan-beam Doppler scatterometer, QuikSCAT/SeaWinds is a conically scanning pencil-beam scatterometer. A pencil-beam scatterometer has several key advantages over a fan-beam scatterometer; It has a higher signal-to-noise ratio, is smaller in size, and provides superior coverage.

The instrument has a carrier frequency of 13.402 GHz (Ku-band). This is carefully chosen to maximize the upwind-downwind asymmetry in ocean scattering signature. The scatterometer transmits a sequence of RF pulses (5.4 msec interval) and measures the returned energy from the Earth surface which is corrupted by the instrument thermal noise (signal-plus-noise energy). In order to estimate an accurate σ_0 , it is necessary to extract the signal-only energy by subtracting the noise-only energy from the signal-plus-noise energy.

The approach of the QuikSCAT/SeaWinds design is to measure the signal-plus-noise energy simultaneously in two receiving channels (also refer to as filters): echo and noise. The instrument periodically performs a calibration measurement sequence in which the transmit power into the receiver through loop-back calibration is measured for radiometric calibration and a cold load is measured for thermal noise estimation. The cold load measurements relate echo channel noise-only energy to the noise channel noise-only energy. This reduces the number of unknown variables to signal-only energy and noise-only energy of the echo channel, which can be solved by two independent signal-plus-noise energy measurements. The noise filter bandwidth is 1 MHz and the echo filter bandwidth can vary from about 5.5 KHz to 198.6 KHz.

Signal Processing for a High Resolution

The QuikSCAT/SeaWinds instrument achieves a resolution higher than the antenna footprint by transmitting a linear frequency-modulated chirp (LFMC) signal. The returned signal

can be viewed as a superposition of scattering contribution from every point in the area illuminated by the antenna beam. Each contributing source is a replica of the transmitted waveform but delayed by a slant range difference which can be related to the time delay. The summed waveform, (i.e. the echo signal), is then multiplied by a LPMC reference signal with a conjugate phase. This step maps a distinct time delay of a received waveform to a corresponding baseband frequency shift in the echo filter.

The extent of the baseband frequency shift allowed depends on the difference of the two commanded parameters: pulse width and echo gate width. The echo gate width - pulse width difference is referred to as the effective gate width which determines the resolution. By design, the echo filter is divided into 12 sub-bands. The center 10 are equally spaced while the 2 outer edge sub-bands are wider. The inner 10 sub-bands are called slices. Figure 1, illustrates the sub-divisions of the echo filter and Table I summarizes the effective gate width modes (a.k.a. slice resolution mode).

TABLE I: QuikSCAT/SeaWinds SES Resolution Mode (KHz)

| Eff. GateWidth (msec) | BW of Mid 10 | BW of Edge 2 | Significance |
|-----------------------|--------------|--------------|--------------|
| 0 | 4.61833 | 0.92 | |
| 0.1 | 23.0916 | 115.48 | |
| 0.2 | 46.1833 | 115.48 | |
| 0.3 | 69.2749 | 92.38 | |
| 0.4 | 129.313 | 69.29 | SES Default |
| 0.5 | 83.1299 | 92.38 | Nominal |
| 0.6 | 106.221 | 78.52 | CDS Default |

Radar Equation and X_factor

The radar equation, which relates the radar return energy in terms of the transmitted energy, the σ_0 , and various electrical and geometric factors, is the basic relation used to compute σ_0 . The critical problem is to extract σ_0 , given the other parameters, by inverting the radar equation.

The radar equation consists of an integral over the scattering surface where the antenna gain, the receiver response, the slant range vary from point to point. As a first approximation, if σ_0 is assumed to be constant over this area (essentially, a mean value), then the σ_0 can be solved as a function of the return/transmit energy ratio and the gain-slant range integral. To avoid cumbersome numerical integration for every pulse in L1B processing, the integration is pre-computed and tabulated based on nominal orbit parameters and antenna measurement geometry. In addition, a perturbation table which can compensate for the errors in the integration due to the difference between the nominal and the actual geometric parameters (orbit and antenna geometry) is pre-computed. The combined nominal and perturbation table value gives the integrated part of the X_factor, X_int. The calibration term of the X_factor, X_cal, together with X_int gives a complete conversion factor to relate the received energy to σ_0 . This table-

driven processing approach is very accurate and significantly reduces L1B processing time. These tables are a function of beam number, antenna azimuth angle, orbit position, slice number, and resolution mode.

Parameters such as slant range and cell area are computed in the Geometry algorithm module. In this sigma0 module, the ratio of echo return energy to transmit energy is estimated from the down-linked QuikSCAT/SeaWinds telemetry parameters, power_DN, noise_DN, and the loopback calibration and cold load measurements. In addition to calculating sigma0, the other main functions of this module yield the signal to noise ratio (SNR) of each measurement and the parameters which determine “communication noise” (Kpc).

Requirement:

Sigma0 processing error must be less than 0.1 dB.

II. FUNCTIONAL FLOW DESCRIPTION

The functional flow of this algorithm module is depicted in Figures 2 and 3. The Sigma0 and Kp algorithm is comprised of four submodules, referred to here by the names of Initialize Calibration Parameters, Get Calibration Parameters, Energy Detection, Sigma0 and Variances Estimation. The primary functions performed by these submodules are described in the following:

Process_Calibration_Data (L1B.3.1.1)

This submodule pre-averages the instrument calibration data and repackages it in a data structure for use in other L1B algorithms. This submodule is called by Execute_Level1B_Algorithms (L1B.3.0.0), the level 1B driver software.

Compute_Sigma0_and_Kp (L1B.3.2.1)

This is the driver algorithm for the sigma0 and Kp module. This algorithm is called by the Level 1B processor Execute_Level1B_Algorithms (L1B.3.0.0).

Get_Cal_Data (L1B.3.3.1)

Given an input pulse time, this submodule retrieves the pre-averaged calibration parameters from a time-ordered data structure created in Process_Calibration_Data (L1B.3.1.1).

Est_Calibration_X (L1B.3.3.2)

Estimates the total calibration factor, X_cal The calibration factor is derived from the calibration measurements, the system loss factors, the instrument parameters, and the antenna peak power.

Calculate_Pr_Pt_Ratio (L1B.3.4.1)

This is a mini-driver algorithm which controls the data flow from the subroutines that calculate signal-only energy, noise-only energy, and their ratio.

Est_Noise_Energy (L1B.3.4.2)

This algorithm submodule calculates the noise-only energy in the echo filter using the power detection equation.

Est_Signal_Energy (L1B.3.4.3)

Calculates the signal-only energy in the echo filter by subtracting out the noise-only energy from the signal-plus-noise energy measurement.

Est_SNR (L1B.3.4.4)

Estimates the signal to noise energy ratio.

Est_Pr_Pt_Ratio (L1B.3.4.5)

Calculates the ratio of received signal-only energy to the transmitted energy.

Calculate_X_Factor (L1B.3.5.1)

Calculates the total correction factor. The correction factor is constructed from the calibration part and the geometry part.

Calculate_Sigma0 (L1B.3.5.2)

Calculates the normalized backscattering coefficient, sigma0.

Calculate_Kpc_Coeff (L1B.3.5.3)

Interpolates the instrument related parameter kpc_A from a pre-calculated table or it calculates Kpc_A for a given instrument operation mode.

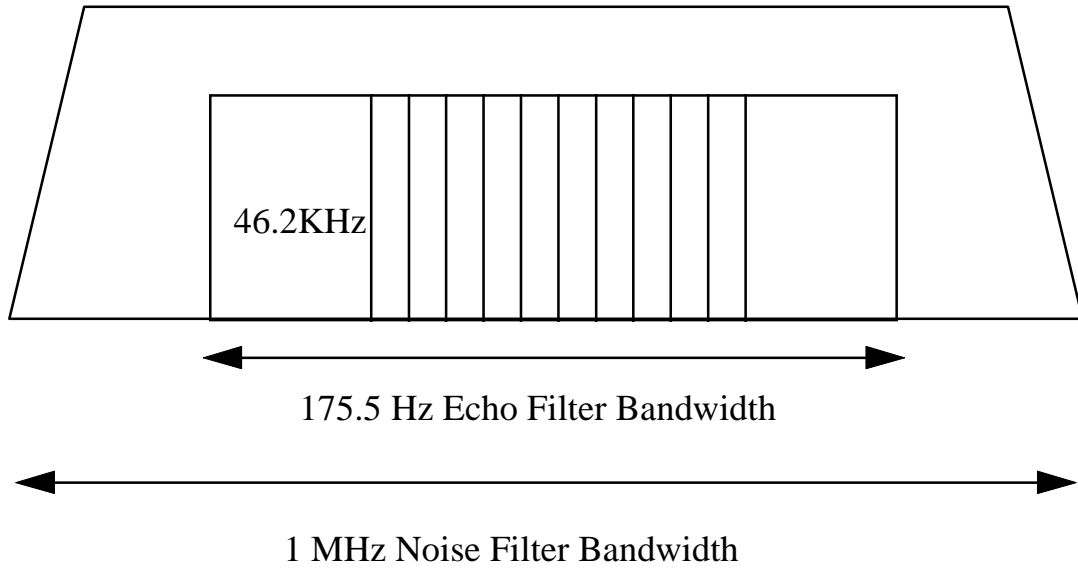


Figure 1: Illustration of Slice Resolution for 0.5msec Effective Gate Width

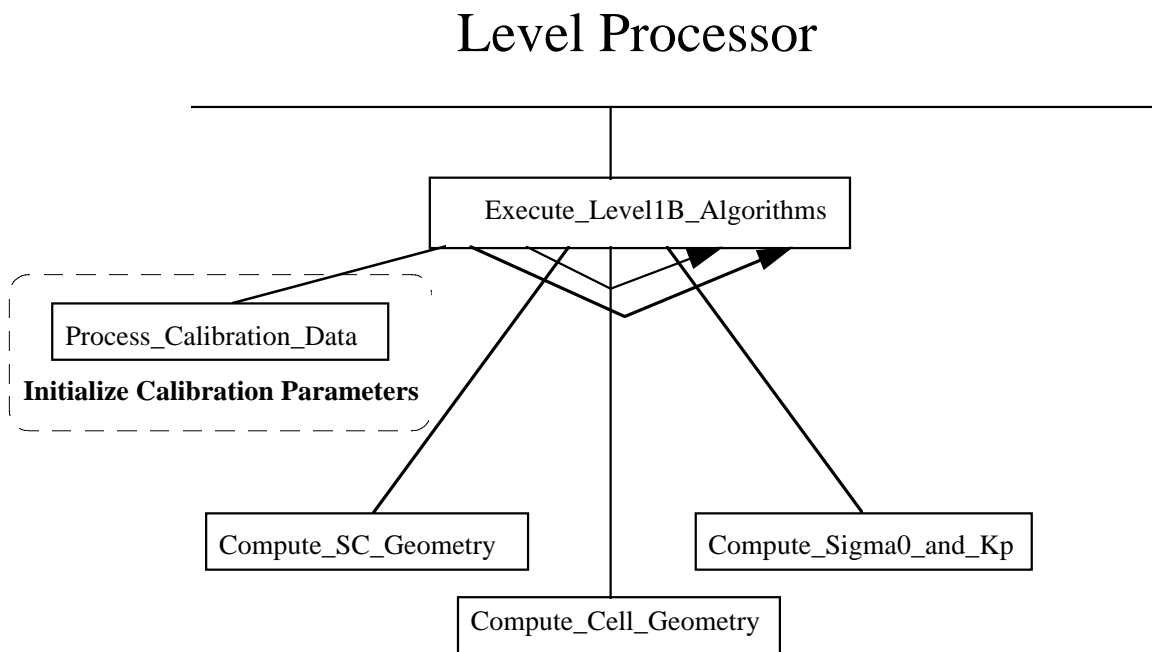


Figure 2: Flow Diagram of `Execute_Level1B_Algorithms` Module

Execute_Level1B_Algorithms

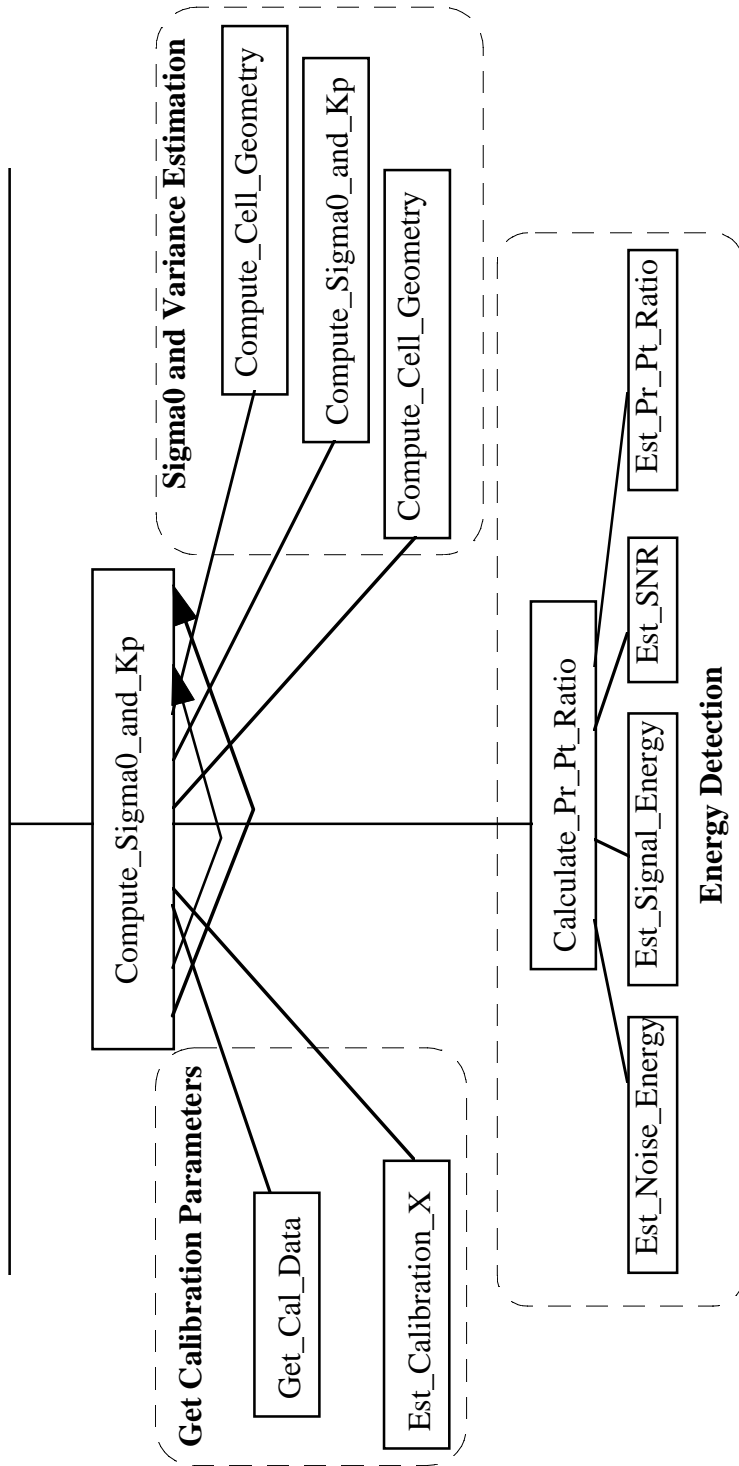


Figure 3: Flow Diagram of Compute_Sigma0_and_Kp

SeaWinds Algorithm Specification

TITLE: Compute the Radar Backscattering Coefficient
SUBMODULE: Initialize Calibration Parameters
MODULE: SeaWinds Sigma0 and Kp Algorithm
CODE: L1B.3.1.1
VERSION: 2.0
DATE: 10/31/99
AUTHOR: Kyung Pak
SUBROUTINE: Process_Calibration_Data
LANGUAGE: FORTRAN
HERITAGE: None

L1B.3.1.1 Process_Calibration_Data

PURPOSE

Pre-average the necessary calibration measurements; bandwidth ratio and signal-only energy for both beams and stored them in a data structure (Cal_Struct) for later use.

BACKGROUND

QuickSCAT/SeaWinds instrument measures the signal-plus-noise energy in both echo and noise filters. Nominally, a measurement is taken every 5.4 msec. By design, the SeaWinds instrument periodically performs a calibration measurement sequence in which the transmit power into the receiver through loop-back calibration is measured for radiometric calibration and a cold load is measured for thermal noise estimation [1]. In order to obtain the backscattering coefficient, sigma0, signal-only energy for the science measurement and the loop-back calibration measurement must be calculated [2,3].

The signal-only energy can be estimated by solving the following power detection equations:

$$P_e = P_s + N_e \quad (1)$$

$$P_n = \beta P_s + \beta \alpha N_e \quad (2)$$

where β is a gain ratio of noise filter to echo filter (a pre-launch calibrated constant of 4.65 dB)

$$\beta \equiv \frac{G_{r,n}}{G_{r,e}} \quad (3)$$

α is a bandwidth ratio of noise filter to echo filter

$$\alpha \equiv \left(\frac{1}{\beta} \right) \frac{\langle P_{n,load} \rangle}{\langle P_{e,load} \rangle} \quad (4)$$

$G_{r,n}$ = noise filter gain.

$G_{r,e}$ = echo filter gain.

P_e = signal-plus-noise energy in the echo filter.

N_e = noise-only energy in the echo filter.

P_s = signal-only energy in the echo filter.

$P_{n,load}$ = cold load calibration measurement in the noise filter.

$P_{e,load}$ = cold load calibration measurement in the echo filter.

and $\langle * \rangle$ represents ensemble average. Solving for the noise-only energy in the echo filter gives

$$N_e = \left(\frac{1}{1-\alpha} \right) \left(P_e - \frac{P_n}{\beta} \right) \quad (5)$$

The signal-only energy can be obtained by equation (1). Similarly, the signal-only energy for a loop-back calibration measurement is

$$P_{s,cal} = \left(\frac{1}{\alpha-1} \right) \left(\alpha \langle P_{e,cal} \rangle - \frac{\langle P_{n,cal} \rangle}{\beta} \right) \quad (6)$$

where

$P_{n,cal}$ = loop-back calibration measurement in the noise filter.

$P_{e,cal}$ = loop-back calibration measurement in the echo filter.

In ground processing the L1A processor extracts loop-back calibration and cold load measurements, system temperature, and corresponding measurement times from level 0 data outputs them to a file. Thus, in L1B processing the complete calibration data- set for a rev is available. The analysis of instrument test data has demonstrated that in equation (4) the cold load measurement must be averaged over at least 800 pulses and in equation (6) the loop-back calibration measurements must be averaged over at least 10 pulses for stability [3]. We use a simple moving average in this algorithm. For QuikSCAT/SeaWinds processing, α and $P_{s,cal}$ values are pre-averaged, time-tagged, and stored in a data structure (Cal_Struct). This provides an efficient time-based access to the calibration data in subsequent processing. This algorithm submodule performs the pre-averaging of the calibration parameters for later use.

REQUIREMENT

The retrieval algorithm must be able to process calibration files with gaps in the data and bad calibration pulses [4].

PROCESSING

Step 1: Given the input `first_frame_time` find a reference index `i_mid`.

Step 2: Find 800 nearest cold load times to the `frame_time(i_mid)` to get `i_load_begin` and `i_load_end`. Also, find 10 nearest loop-back calibration pulse times to the `frame_time(i_mid)` to get `i_cal_a_begin`, `i_cal_a_end`, `i_cal_b_begin`, and `i_cal_b_end` (for the loop-back calibration pulses, the inner/outer beam are pre-averaged separately [5]).

Step 3: Calculate the quantities $\langle P_{n,load} \rangle$, $\langle P_{e,load} \rangle$, $\langle P_{n,cal} \rangle$, and $\langle P_{e,cal} \rangle$.
`frame_time_cal_secs = frame_time(i_mid)`
`num_cal_recs = num_cal_recs + 1`

Step 4: Calculate bandwidth ratio and loop-back calibration signal-only energy using equations (4) and (6), respectively. Store the pre-averaged calibration data: `frame_time_cal_secs`, `bandwidth_ratio`, and `loopback_cal_A/B`.

Step 5: Get the next `frame_time` and increment `i_mid` if necessary. Repeat Steps 1-4 until the end of the calibration data file. Finally, assign the `num_cal_recs` in the calibration data structure.

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|-------------------------------|--------------|---|
| <code>first_frame_time</code> | sec | First frame time in a rev. |
| <code>cal_pulse_struct</code> | | Data structure containing one full set of calibration pulse data for the rev. |

OUTPUT

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|-------------------------|--------------|---|
| <code>Cal_Struct</code> | | This data structure contains <code>num_cal_recs</code> pre-averaged calibration data. It contains the corresponding <code>frame_time_cal_secs</code> , <code>bandwidth_ratio</code> , and <code>loopback_cal_A/B</code> . |

REFERENCES

[1] Liu, Yong, "QuikSCAT Telemetry Packet Processing - Order of Science Measurements," IOM 3347-98-029, June 7, 1998.

[2] [Lou, Shu-Hsiang and Yong Liu, "SeaWinds/QuikSCAT Sigma0 Calibration Model," IOM 3347-97-025, December 19, 1997.

- [3] Lou, Shu-Hsiang and Yong Liu, “SeaWinds/QuikSCAT High and Low Resolution On-orbit Calibration and Noise,” IOM 3347-98-019, March 27, 1998.
- [4] Liu, Yong, “QuikSCAT Telemetry Packet Processing - Approach for Eliminating Bad Calibration Measurement”, IOM 3347-98-030, June 8, 1998.
- [5] Liu, Yong, “Loop-back Calibration Pulse Processing in SES,” IOM 3347-98-010, January 30, 1998.

SeaWinds Algorithm Specification

TITLE: Compute the Radar Backscattering Coefficient
SUBMODULE: Cell's Sigma0 Estimation
MODULE: SeaWinds Sigma0 and Kp Algorithm
CODE: L1B.3.2.1
VERSION: 2.0
DATE: 10/31/99
AUTHOR: Kyung Pak and Angela Zhang
SUBROUTINE: Compute_Sigma0_and_Kp
LANGUAGE: FORTRAN
HERITAGE: None

L1B.3.2.1 Compute_Sigma0_and_Kp

PURPOSE

This driver algorithm computes the radar backscattering coefficient (sigma0), the signal to noise ratio, and the normalized standard deviation of sigma0 (Kpc) due to communication noise.

BACKGROUND

This algorithm drives the Sigma0 and Kp module. It first calls Get_Cal_Data (L1B.3.3.1) to retrieve the pre-averaged calibration pulse measurements corresponding to the pulse time. Then it calls Est_Calibration_X (L1B.3.3.2) to estimate the calibration part of the X-factor using the loss factors and the retrieved calibration pulse measurements. It next calls Calculate_Pr_Pt_Ratio (L1B.3.4.1) to calculate the signal energy, noise energy, ratio of the signal energy to the noise energy, and the ratio of the signal energy to the transmitted energy. Then it calls Calculate_X_Factor (L1B.3.5.1) to return the total X-factor, and calls Calculate_Sigma0 (L1B.3.5.2) to calculate the sigma0 value. Finally, it calls Calculate_Kpc_Coeff (L1B.3.5.3) to calculate the instrument related parameter Kpc_A.

REQUIREMENT

The sigma0 processing error must be less than 0.1 dB.

PROCESSING

First, the processor checks the pulse quality to determine whether to proceed forward with the execution of the sigma0 and Kp algorithms. Then the processor initializes parameters needed by the sigma0 and Kp module and its subroutines: receiver index, effective gate width, and beam number. Once the initialization is completed, the processor proceeds by calling the following

algorithms in sequence:

Step 1: Retrieve pre-averaged loopback calibration and load pulse data associated with the current pulse time.

call Get_Cal_Data (L1B.3.3.1)

Step 2: Estimate calibration components of the X-factor; X_Cal_A and X_Cal_B.

call Est_Calibration_X (L1B.3.3.2)

Step 3: Calculate energy ratios Pr/Pt and SNR.

call Calculate_Pr_Pt_Ratio (L1B.3.4.1)

Step 4: Calculate total X_factor values.

call Calculate_X_Factor (L1B.3.5.1).

Step 5: Calculate Sigma0 to estimate the radar backscattering coefficient.

call Calculate_Sigma0 (L1B.3.5.2)

Step 6: Estimate the normalized standard deviation coefficients.

call Calculate_Kpc_Coeff (L1B.3.5.3)

CONSTANTS/TABLES

| <u>Constant Name</u> | <u>Units</u> | <u>Description</u> |
|------------------------------|--------------|---|
| Global_Constants | | Constants such as p. |
| L1B_Constants | | The run time constants required by the Level 1B Processor. |
| Cal_Struct | | This data structure contains num_cal_recs pre-averaged calibration data. It contains the corresponding frame_time_cal_secs, bandwidth_ratio, and loopback_cal_A/B. |
| Topo_Height | m | Earth topography map table. |
| S_Table | 1/m | A correction coefficient table used to compensate for a baseband frequency error. |
| X_Factor_Table | $(1/m)^2$ | A table containing the geometric factor in the radar equation that is based on the calculation of an integral. |
| cal_temperature_coefficients | | An array of coefficients in a polynomial expansion of fourth or lower order in spacecraft temperature to determine the overall loss factor in the calibration loop between the SeaWinds transmitter and receiver. |
| slice_noise_fraction | | A table of multipliers which, when applied to the estimated pulse echo noise energy, generates a noise energy for one of the slices within the echo signal. |

| | |
|------------------|---|
| bb_freq_off_corr | A correction to the calculated change in baseband frequency. This correction reflects biases in the baseband frequency which are due to perturbations in the spacecraft attitude and the antenna look angles. |
| Kpc_Parameters | Coefficients which are used to determine the normalized standard of the sigma0 measurement due to the statistical nature of the received signal, or kpc_A. |

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|---------------------------|--------------|--|
| sigma0_mode_flag | | Bit flag which indicate the instrument mode and status at the time the measurement was acquired. |
| frame_time_sec | sec | Frame time. |
| pulse_time | sec | Pulse time. |
| orbit_time | counts | Current orbit time. |
| rev_orbit_period | sec | Time required for the spacecraft to complete one complete orbit. |
| precision_coupler_temp_eu | Celsius | Precision coupler temperature. |
| rcv_protect_sw_temp_eu | Celsius | Receive protect switch temperature. |
| beam_select_sw_temp_eu | Celsius | Beam select switch temperature. |
| receiver_temp_eu | Celsius | Receiver temperature. |
| range_gate_A_width | DN | Range gate width for beam A. |
| range_gate_B_width | DN | Range gate width for beam B. |
| pulse_width | DN | Pulse width. |
| antenna_azimuth | deg | Antenna azimuth angle. |
| cell_lat | deg | Geodetic cell latitude. |
| cell_lon | deg | Cell longitude. |
| power_DN | DN | Energy measured by the echo filter. |
| noise_DN | DN | Energy measured by the noise filter. |
| L1B_slice_dim | | Maximum slice dimension. |
| frequency_shift | Hz | The shift in the baseband frequency of a pulse due to errors in the Doppler Binning Table and the spacecraft attitude measurement. |

OUTPUT

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|---|
| bandwidth_ratio | | The ratio of the rolling average of about 800 load calibration measurements through the noise filter to the echo filter |
| slice_SNR | dB | The ratio of signal to noise for a slice. |
| SNR | dB | The ratio of signal to noise for a whole pulse. |

| | | |
|------------------|----|--|
| slice_Kpc_A | | Normalized standard deviation of slice signal. |
| sigma0_Kpc_A | | Normalized standard deviation of the whole echo signal. |
| slice_sigma0 | dB | Radar backscattering coefficient for each slice. |
| sigma0 | dB | Radar backscattering coefficient for a pulse. |
| X_cal_A | dB | Beam A radiometric calibration component of the X-factor. |
| X_cal_B | dB | Beam B radiometric calibration component of the X-factor. |
| X_factor | dB | Complete conversion factor from scatterometer received signal energy to sigma0 value for each slice. |
| slice_qual_flag | | Bit flags which indicate the quality of the data for each slice. |
| sigma0_qual_flag | | Bit flag which indicate the quality of the data for a pulse. |

REFERENCES

- [1] NASA/JPL Scatterometry Processing Algorithm and Analysis Group, "Science Algorithm Specification for SeaWinds", JPL, October 20, 1996.

SeaWinds Algorithm Specification

TITLE: Compute the Radar Backscattering Coefficient
SUBMODULE: Get Calibration Parameters
MODULE: SeaWinds Sigma0 and Kp Algorithm
CODE: L1B.3.3.1
VERSION: 2.0
DATE: 10/31/99
AUTHOR: Kyung Pak
SUBROUTINE: Get_Cal_Data
LANGUAGE: FORTRAN
HERITAGE: None

L1B.3.3.1 Get_Cal_Data

PURPOSE

For a given input time (pulse_time), Get_Cal_Data.F retrieves the necessary pre-averaged loop-back calibration elements, bandwidth_ratio, Signal_energy_cal_A and Signal_energy_cal_B from a data structure (Cal_Struct).

BACKGROUND

Some of the instrument parameters needed to calculate sigma0 varies with time and/or temperature on orbit. Therefore, it is necessary to monitor these changes in the instrument continuously. By design, the SeaWinds instrument frequently performs a calibration measurement sequence in which the transmit power into the receiver through loop-back calibration is measured for radiometric calibration and a cold load is measured to estimate the thermal noise. In ground processing the L1A processor extracts loop-back calibration and cold load pulses, system temperature, and corresponding measurement times from the telemetry and outputs them to a file. In Process_Calibration_Data.F (L1B.3.1.1), an algorithm averages over 800 load pulses and 10 loop-back calibration pulses. Then they are stored in a data structure, Cal_Struct. The retrieved variables are used in the power detection algorithm in Calculate_Pr_Pt_Ratio.F

REQUIREMENT

The retrieval algorithm must be able to process files with gaps in the data and bad calibration pulses [1].

PROCESSING

Step 1: Given the input pulse_time, perform a simple search by comparing the pulse_time and the calibration time.

```
DO WHILE ( (.NOT. found_it) .AND. (.NOT. end_of_file) )
  IF (pulse_time .LT. Cal_Struct.averages(index).frame_time_cal_secs) THEN
    found_it = .TRUE.
  ELSE
    index = index + 1

    IF (index .GT. Cal_Struct.num_cal_recs ) THEN
      end_of_file = .TRUE.
    END IF

  END IF
END DO
```

Step 2: Assign the appropriate signal_energy_cal and bandwidth_ratio values from the file using the index variable from Step 1. Also, assign the new_cal flag by comparing current index with the index_old.

```
IF (.NOT. end_of_file) THEN

  IF (index .GT. index_old) THEN
    signal_energy_cal_A_save = Cal_Struct.averages(index).loop_back_cal_A
    signal_energy_cal_B_save = Cal_Struct.averages(index).loop_back_cal_B
    bandwidth_ratio_save = Cal_Struct.averages(index).bandwidth_ratio
    new_cal = .TRUE.
    index_old = index
  ELSE IF (index .EQ. index_old) THEN
    new_cal = .FALSE.
  END IF

  signal_energy_cal_A = Signal_energy_cal_A_save
  Signal_energy_cal_B = Signal_energy_cal_B_save
  bandwidth_ratio = bandwidth_ratio_save

END IF
```

CONSTANTS/TABLES

| <u>Name</u> | <u>Description</u> |
|---------------|--|
| L1B_Constants | The run time constants required by the Level 1B Processor. |

| | |
|------------|--|
| Cal_Struct | This data structure contains num_cal_recs pre-averaged calibration data. It contains the corresponding frame_time_cal_secs, bandwidth_ratio, and loopback_cal_A/B. |
|------------|--|

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|--------------------|
| pulse_time | sec | Pulse time. |

OUTPUT

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|------------------------|--------------|---|
| bandwidth_ratio | | The ratio of the rolling average of about 800 load calibration measurements through the noise filter to the echo filter |
| Signal_energy_cal_A DN | | Calibration energy of beam A pulse. |
| Signal_energy_cal_B DN | | Calibration energy of beam B pulse. |
| new_cal | | A flag indicating whether the signal_energy_cal_A and signal_energy_cal_B was updated. |

REFERENCES

- [1] Liu, Yong, "QuikSCAT Telemetry Packet Processing - Approach for Eliminating Bad Calibration Measurements", IOM 3347-98-030, June 8,1998.

SeaWinds Algorithm Specification

TITLE: Compute the Radar Backscattering Coefficient
SUBMODULE: Get Calibration Parameters
MODULE: SeaWinds Sigma0 and Kp Algorithm
CODE: L1B.3.3.2
VERSION: 2.0
DATE: 10/31/99
AUTHOR: Kyung Pak
SUBROUTINE: Est_Calibration_X
LANGUAGE: FORTRAN
HERITAGE: None

L1B.3.3.2 Est_Calibration_X

PURPOSE

For a given signal-only energy of the loop-back calibration pulse, determine the calibration part of the X-factor for both beams.

BACKGROUND

A backscattering coefficient, sigma0, and a signal-only energy measurement, Ps, is related by the X-factor. The X-factor is a normalization factor which can be separated into two parts; Xcal and Xint

$$\sigma_0 = P_s [X_{cal} X_{int}]^1 \quad (1)$$

where

$$X_{cal} \equiv \frac{\lambda^3}{(4\pi)^3} G_p^2 \left(\frac{1}{L_a^2 L_w^2} \right) \left[\frac{L_{bias} L_{23}}{L_{13} L_{21}} \frac{L_{cal}}{L_{op}} \right] P_{s_cal} \quad (2)$$

$$X_{int} \equiv \iint d^2 r \frac{F(f_b(r))}{R^4(r)} \quad (3)$$

where

λ = Wavelength of the transmitted pulse.
 G_p = Antenna peak gain.
 P_{s_cal} = Signal-only energy of the loop-back calibration pulse.
 P_s = Signal-only energy for an entire footprint in the echo channel.

| | |
|-----------------|---|
| $F(f_b(r))$ | = Digital filter response of an entire footprint. The antenna gain pattern is also absorbed in this term. |
| $PtGr$ | = Transmitted power-receiver gain product. |
| $\sigma_0(x,y)$ | = Scattering coefficient at position (x,y) on surface. |
| $R(r)$ | = Slant range to a point (x,y) on surface. |
| L_{bias} | = Bias between the modeled and measured loss factor (-1.25 dB for receiver A and -1.17 dB for receiver B) |
| L_w | = One way platform waveguide loss (0.21 dB inner beam, 0.24 dB outer beam). |
| L_a | = Atmospheric loss. |

and

$$L_{sys} = \left[\frac{L_{bias} L_{13} L_{21}}{L_{23}} \frac{L_{op}}{L_{cal}} \right] \quad (4)$$

where the loss factor terms are

| | |
|------------------|-------------------------------------|
| L_{13} | = Receiver path insertion loss. |
| L_{21} | = Transmitter path insertion loss. |
| L_{23} | = Calibration path insertion loss. |
| L_{cal}/L_{op} | = Calibration path attenuator loss. |

Similar expression exists for the slice σ_0 . The X_{cal} consists of system loss, calibration constants, and signal-only energy of the loop-back calibration. The loss factors are functions of system temperatures. To continuously track the varying system temperatures and P_{s_cal} , they are periodically monitored and reported in the telemetry. For a given temperature, the loss factors are modeled as N-th order polynomial expansion:

$$L = \sum_{n=0}^N A_n T^n \quad (5)$$

The polynomial expansion coefficients, A_n , are stored in a data structure array `cal_temperature_coefficients`. Also, `cal_temperature_coefficients` stores the high and low temperature range values where these polynomial expansions are applicable. They range from 0 to 55 degrees Celsius. The following table lists the complete temperature ranges:

Loss Factor Temperature Range Table

| Loss Factor | Low (Celsius) | High (Celsius) |
|---------------------------|---------------|----------------|
| L_{13} | 0 | 55 |
| L_{21} | 0 | 55 |
| L_{23} | 0 | 55 |
| L_{cal}/L_{op} (side A) | 5 | 40 |
| L_{cal}/L_{op} (side B) | 5 | 40 |

In processing, for those temperatures values outside the valid ranges, the temperatures are “pinned” to the min/max values and used in the loss factor calculations. The processor indicates that the temperature values are out of range by turning on the appropriate bit in the sigma0_qual_flag . When this bit is turned on the sigma0 must be used with caution.

The evaluation of the X_{int} term is not computed by the L1B processor. Instead, it is pre-computed and tabularized based on expected orbit and measurement geometry. The X_{int} table is a function of operational mode, orbit position, antenna azimuth angle, beam number, and slice number.

REQUIREMENT

The sigma0 processing error must be less than 0.1 dB.

PROCESSING

Step 1: Given input temperatures, precision_coupler_temp_eu, rcv_protect_sw_temp_eu, beam_select_sw_temp_eu, and receiver_temp_eu, compare the old and the new temperatures. If the new set of temperatures are different than the old set of temperatures, then proceed to check for the temperature ranges. If the new set of temperatures are the same as the old set, then set skip to the end of the subroutine.

Step 2: Check the temperature ranges. If they are less than LOSS_FACTOR_HIGH_TEMP and greater than LOSS_FACTOR_LOW_TEMP then proceed with loss factor calculation. For temperatures outside the range, “pin” the temperature to the appropriate limit and turn on the temperature range bit flag of the sigma0_qual_flag, then proceed.

Step 3: Calculate each loss factors using equation (4).

Step 4: Calculate the total system loss factor (all the loss terms in equation (2)) and the X_{cal_A} and X_{cal_B} using equation (2).

IF (old_temp .NE. new_temp) THEN

IF (new_temp .LT. max_temp_range .AND.
new_temp .GT. min_temp_range) THEN

temp_within_range = .TRUE.

ELSE IF (new_temp .GT. max_temp_range) THEN
temp_within_range = .FALSE.

temp = max_temp_range

ELSE IF (new_temp .LT. min_temp_range) THEN

```

temp_within_range = .FALSE.
temp = min_temp_range
END IF

```

```

calculate the loss factors
calculate X_cal_A and X_cal_B

```

```

END IF

```

CONSTANTS/TABLES

| <u>Name</u> | <u>Units</u> | <u>Description</u> |
|------------------------------|--------------|--|
| Global_Constants | | Global Constants such as p. |
| L1B_Constants | | The run time constants required by the Level 1B Processor. |
| cal_temperature_coefficients | | The array lists the coefficients in a polynomial expansion of fourth or lower order in spacecraft temperature. The loss_factor_high_temp array and the loss_factor_low_temp arrays specify the range of temperatures where these polynomial expansions are applicable. |

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|---------------------------|--------------|--|
| receiver_index | | An identifier which indicates whether SES side A/B is active. |
| signal_energy_cal_A | DN | The signal-only energy in the echo channel for calibration beam A. |
| signal_energy_cal_B | DN | The signal-only energy in the echo channel for calibration beam B. |
| precision_coupler_temp_eu | Celsius | The temperature of the precision coupler. |
| rcv_protect_sw_temp_eu | Celsius | The temperature of the receive protect switch. |
| beam_select_sw_temp_eu | Celsius | The temperature of the beam select switch. |
| receiver_temp_eu | Celsius | The temperature of the receiver. |

OUTPUT

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|---|
| temp_within_range | | A logical flag which indicate if the temperature is within a specified range. |
| L_sys | dB | The SES calibration loss factor. |

| | | |
|-----------------|------------------------------|---|
| X_cal_A/X_cal_B | $\text{DN} \cdot \text{m}^2$ | The beam A/B component of the X-factor which includes the system and calibration loss factors, the antenna gain, as well as the loop-back calibration pulse measure |
|-----------------|------------------------------|---|

REFERENCES

- [1] Lou, Shu-Hsiang, "QuikSCAT Loss Factors," IOM 3347-98-001, January 15, 1998.
- [2] Lou, Shu-Hsiang, "QuikSCAT/SeaWinds X/K Factor Formulation," IOM 3347-98-008, January 26, 1998.
- [3] Lou, Shu-Hsiang, "SeaWinds Loss Factors," IOM 3347-98-050, August 20, 1998.
- [4] Liu, Yong, "QuikSCAT Pcal Bias and Its Removal," IOM 3347-98-056, October 5, 1998.

SeaWinds Algorithm Specification

TITLE: Compute the Radar Backscattering Coefficient
SUBMODULE: Energy Detection
MODULE: SeaWinds Sigma0 and Kp Algorithm
CODE: L1B.3.4.1
VERSION: 2.0
DATE: 03/03/99
AUTHOR: Kyung Pak
SUBROUTINE: Calculate_Pr_Pt_Ratio
LANGUAGE: FORTRAN
HERITAGE: None

L1B.3.4.1 Calculate_Pr_Pt_Ratio

PURPOSE

This is a mini-driver which controls the processing calls to Est_Noise_Energy.F, Est_Signal_Energy.F, Est_SNR.F, and Est_Pr_Pt_Ratio.F.

BACKGROUND

QuickSCAT instrument measures the signal-plus-noise energy in both echo and noise channels. In order to obtain the backscattering coefficient, sigma0, noise-only energy must be estimated and subtracted from the signal-plus-noise energy. Calculate_Pr_Pt_Ratio.F is a mini-driver subroutine. Its main function is to make sequential FORTRAN calls to Est_Noise_Energy.F to estimate the noise-only energy, Est_Signal_Energy.F to estimate the signal-only energy, Est_SNR.F to calculate the ratio of the signal-only to noise-only energy, and Est_Pr_Pt_Ratio.F to calculate the ratio of the signal-only energy to the transmitted energy. Then it passes the necessary variables to the calling routine (Compute_Sigma0_and_Kp.F) for subsequent sigma0 processing.

Calculate_Pr_Pt_Ratio is executed under both Wind Observation Mode (WOM) and Receive Only Mode (ROM). In WOM, the mini-driver calls all four algorithms to estimate noise-only energy, signal-only energy, SNR and Pr_Pt_ratio. In ROM, it calls only the core power detection algorithm subroutines Est_Noise_Energy.F and Est_Signal_Energy.F.

PROCESSING

Step 1: Estimate the noise-only energy for each slice:

CALL Est_Noise_Energy

Step 2: Estimate the signal-only energy for each slice:

CALL Est_Signal_Energy

Step 3: If the current instrument mode is WOM, then estimate the SNR. Otherwise skip the rest of the subroutine.

CALL Est_SNR

Step 4: If the current instrument mode WOM, then estimate the Pr_Pt_ratio. Otherwise skip the rest of the subroutine.

CALL Est_Pr_Pt_Ratio

CONSTANTS/TABLES

| <u>Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|--|
| L1B_Constants | | The run time constants required by the Level 1B Processor. |
| slice_noise_fraction | | Factors used to calculate the noise-only energy for each of the slices in the echo filter. |

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|---|
| bandwidth_ratio | | The ratio of the rolling average of about 800 load calibration measurements through the noise filter to the echo filter |
| Signal_energy_cal_A | DN | Calibration signal-only energy of beam A pulse. |
| Signal_energy_cal_B | DN | Calibration signal-only energy of beam B pulse. |
| power_DN | DN | Energy measured by the echo filter. |
| noise_DN | DN | Energy measured by the noise filter. |
| L_sys | dB | The SES calibration loss factor. |
| mode_index | | An identifier indicating the current effective gate width (1-8). |
| receiver_index | | An identifier which indicates whether SES side A/B is active. |
| L1B_slice_dim | | Maximum slice dimension. |
| sigma0_mode_flag | | The bit flag which indicate the instrument mode and status at the time the measurement was acquired. |

OUTPUT

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|--|
| slice_qual_flag | | Bit flags which indicate the quality of the data for each slice. |
| sigma0_qual_flag | | Bit flag which indicate the quality of the data for a pulse. |

| | | |
|---------------------|----|--|
| slice_SNR | | The ratio of signal to noise for a slice. |
| SNR | | The ratio of signal to noise for a whole pulse. |
| slice_Pr_Pt_ratio | | The ratio of the signal-only energy to the transmit energy for each slice. |
| Pr_Pt_ratio | | The ratio of the sum of signal-only energy of center 10 slices to the transmit energy. |
| signal_energy_slice | DN | Signal-only energy of the echo filter output measurement for each slice |
| signal_energy | DN | Sum of signal-only energy of the echo filter measurements for the center 10 slices. |

REFERENCE

- [1] NASA/JPL Scatterometry Processing Algorithm and Analysis Group, "Science Algorithm Specification for SeaWinds", JPL, October 20, 1996.

SeaWinds Algorithm Specification

TITLE: Compute the Radar Backscattering Coefficient
SUBMODULE: Energy Detection
MODULE: SeaWinds Sigma0 and Kp Algorithm
CODE: L1B.3.4.2
VERSION: 2.0
DATE: 10/31/99
AUTHOR: Kyung Pak
SUBROUTINE: Est_Noise_Energy
LANGUAGE: FORTRAN
HERITAGE: None

L1B.3.4.2 Est_Noise_Energy

PURPOSE

This subroutine calculates the noise-only energy in the echo filter in data number units (DN).

BACKGROUND

QuickSCAT/SeaWinds instrument measures the signal-plus-noise energy in both echo and noise filters. In order to obtain the backscattering coefficient, sigma0, noise-only energy of the echo filter must be estimated and subtracted from the signal-plus-noise energy. In order to estimate the received noise-only energy in the echo filter the receiver gain ratio (b) must be determined. b is a gain ratio of noise filter to echo filter, a pre-launch calibrated constant (4.65dB). Also, the bandwidth ratio (a) of noise filter to echo filter needs to be calibrated on orbit.

Equations for the whole pulse:

The noise-only energy can be estimated by solving the following power detection equations:

$$P_e = P_s + N_e \quad (1)$$

$$P_n = \beta P_s + \beta \alpha N_e \quad (2)$$

where

$$\beta \equiv \frac{G_{r,n}}{G_{r,e}} \quad (3)$$

$$\alpha \equiv \left(\frac{1}{\beta} \right) \frac{\langle P_{n,load} \rangle}{\langle P_{e,load} \rangle} \quad (4)$$

$G_{r,n}$ = noise filter gain.
 $G_{r,e}$ = echo filter gain.
 P_e = signal-plus-noise energy in the echo filter.
 N_e = noise-only energy in the echo filter.
 P_s = signal-only energy in the echo filter.
 $P_{n,load}$ = cold load calibration measurement in the noise filter.
 $P_{e,load}$ = cold load calibration measurement in the echo filter.

Solving for the noise-only energy in the echo filter gives

$$N_e = \left(\frac{1}{1-\alpha} \right) \left(P_e - \frac{P_n}{\beta} \right) \quad (5)$$

Once the noise-only energy has been estimated using equation (5), signal-only energy in the echo filter can be obtained using equation (1).

Equations for the individual slices:

The procedure for calculating the signal-only energy for individual slices follow similar steps. For slices the echo filter measurement is

$$P_{e,i} = P_{s,i} + N_{e,i} \quad (6)$$

where the slice bandwidth ratio and a slice noise fraction coefficient, q_i , are defined as

$$\alpha_i \equiv \left(\frac{1}{\beta} \right) \frac{\langle P_{n,load} \rangle}{\langle P_{e,i,load} \rangle} \quad (7)$$

$$q_i \equiv \frac{\langle P_{e,i,load} \rangle}{\langle P_{e,load} \rangle} = \frac{\alpha}{\alpha_i} \quad (8)$$

$P_{e,i,load}$ = cold load calibration measurement for the individual slices in the echo filter.

Using the slice noise fraction coefficient, the individual slice noise-only energy measurement is calculated by the following relation

$$N_{e,i} = q_i N_e \quad (9)$$

Once the $N_{e,i}$ is determined, the individual slice signal-only energy is obtained using equation (6).

PROCESSING

Step 1: Calculate noise-only energy for the whole pulse by using the power detection equation:

```
DO i_slice = 1,num_slices
  echo_energy = power_DN(i_slice) + echo_energy
END DO

echo_noise_energy
= (noise_DN/RECEIVER_GAIN_RATIO- echo_energy) / (bandwidth_ratio - 1)
```

Step 2: Estimate the noise-only energy for the center 8 slices:

```
DO i_slice = 1,L1B_slice_dim
  echo_noise_energy_slice(i_slice) = echo_noise_energy
  * slice_noise_fraction(i_slice + 2,mode_index,receiver_index)
END DO
```

Step 3: Calculate the total signal-plus-noise energy of the center 10 slices and store it in echo_energy:

```
echo_energy = 0.0

DO i_slice = 1, NUM_SLICES_PER_SIGMA0
  egg_noise_fraction = egg_noise_fraction
  + slice_noise_fraction(i_slice + 1,mode_index,receiver_index)
  echo_energy = FLOAT(power_DN(i_slice + 1)) + echo_energy
END DO
```

Step 4: Calculate the total noise-only energy of the center 10 slices:

```
echo_noise_energy = echo_noise_energy*egg_noise_fraction
```

CONSTANTS/TABLES

| <u>Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|--|
| L1B_Constants | | The run time constants required by the Level 1B Processor. |
| slice_noise_fraction | | Factors used to calculate the noise-only energy for each of the slices in the echo filter. |

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|---|
| mode_index | | An identifier indicating the current effective gate width (1-8). |
| receiver_index | | An identifier which indicates whether SES side A/B is active. |
| L1B_slice_dim | | Maximum slice dimension. |
| bandwidth_ratio | | The ratio of the rolling average of about 800 load calibration measurements through the noise filter to the echo filter |
| power_DN | DN | Energy measured by the echo filter. |
| noise_DN | DN | Energy measured by the noise filter. |

OUTPUT

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|-------------------------|--------------|---|
| echo_energy | DN | Sum of signal-plus-noise energy in the center 10 slices in the echo filter. |
| echo_noise_energy_slice | DN | Noise-only energy in each slice in echo filter |
| echo_noise_energy | DN | Sum of Noise-only energy in the center 10 slices in the echo filter. |

REFERENCES

- [1] Lou, Shu-Hsiang and Yong Liu, "SeaWinds/QuikSCAT High and Low Resolution On-orbit Calibration and Noise," IOM 3347-98-019, March 27, 1998.

SeaWinds Algorithm Specification

TITLE: Compute the Radar Backscattering Coefficient
SUBMODULE: Energy Detection
MODULE: SeaWinds Sigma0 and Kp Algorithm
CODE: L1B.3.4.3
VERSION: 2.0
DATE: 10/31/99
AUTHOR: Kyung Pak
SUBROUTINE: Est_Signal_Energy
LANGUAGE: FORTRAN
HERITAGE: Est_Echo_Return_Energy (SeaWinds ATB L1B.3.2.1)

L1B.3.4.3 Est_Signal_Energy

PURPOSE

This subroutine calculates the signal-only energy in the echo filter in data number units (DN).

BACKGROUND

QuickSCAT/SeaWinds instrument measures the signal-plus-noise energy in both echo and noise filters. In order to obtain the backscattering coefficient, sigma0 , noise-only energy of the echo filter must be estimated and subtracted from the signal-plus-noise energy. The noise-only energy has been estimated in Est_Noise_Energy.F (L1B.3.4.2). The signal-only energy of the pulse in the echo filter can be obtained using the following relation

Equation for the whole pulse:

$$P_e = P_s + N_e \quad (1)$$

P_e = signal-plus-noise energy in the echo filter.

N_e = noise-only energy in the echo filter.

P_s = signal-only energy in the echo filter.

Equations for the individual slices:

The procedure for calculating the signal-only energy for individual slices is

$$P_{e,i} = P_{s,i} + N_{e,i} \quad (2)$$

PROCESSING

Step 1: Calculate signal-only energy for the whole pulse by using equation (1):

$$\text{signal_energy} = \text{echo_energy} - \text{echo_noise_energy}$$

Step 2: Calculate the signal-only energy for the center 8 slices using equation (2):

DO i_slice = 1, L1B_slice_dim ! 8 center slices.

$$\begin{aligned} \text{signal_energy_slice}(i_slice) = & \text{power_DN}(i_slice + 2) \\ & - \text{echo_noise_energy_slice}(i_slice) \end{aligned}$$

END DO

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|-------------------------|--------------|---|
| L1B_slice_dim | | Maximum slice dimension. |
| power_DN | DN | Energy measured by the echo filter. |
| echo_energy | DN | Sum of signal-plus-noise energy in the center 10 slices in the echo filter. |
| echo_noise_energy_slice | DN | Noise-only energy in each slice in echo filter |
| echo_noise_energy | DN | Sum of Noise-only energy in the center 10 slices in the echo filter. |

OUTPUT

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|--|
| signal_energy_slice | DN | Signal-only energy in each slice of the echo filter . |
| signal_energy | DN | Sum of the signal-only energy in the center 10 slices. |

REFERENCES

- [1] Lou, Shu-Hsiang and Yong Liu, "SeaWinds/QuikSCAT High and Low Resolution On-orbit Calibration and Noise," IOM 3347-98-019, March 27, 1998.

SeaWinds Algorithm Specification

TITLE: Compute the Radar Backscattering Coefficient
SUBMODULE: Energy Detection
MODULE: SeaWinds Sigma0 and Kp Algorithm
CODE: L1B.3.4.4
VERSION: 2.0
DATE: 10/31/99
AUTHOR: Kyung Pak and Angela Zhang
SUBROUTINE: Est_SNR
LANGUAGE: FORTRAN
HERITAGE: Est_SNR (SeaWinds ATB L1B.3.2.2)

L1B.3.4.4 Est_SNR

PURPOSE

This subroutine calculates the ratio of the signal power to the thermal noise power within the echo detection bandwidth (SNR). The SNR is used to calculate the normalized standard deviation of the echo energy, Kpc.

BACKGROUND

QuickSCAT/SeaWinds radar measures backscattered energy from the earth surface. The measurement is noisy due to instrument thermal noise and signal fading effects. An accurate noise estimation must be made in order to optimize the maximum likelihood estimator-based wind retrieval algorithm. A parameter used to describe the magnitude of the noise is Kpc. This parameter is a function of the signal to noise ratio. The noise-only energy was estimated in Est_Noise_Energy.F (L1B.3.4.2) and the signal-only energy was estimated in Est_Signal_Energy.F (L1B.3.4.3). This subroutine calculates the SNR.

For the whole pulse, the echo channel measures the signal-plus-noise energy

$$P_e = P_s + N_e \quad (1)$$

The ratio of the signal-only energy to the noise-only energy gives

$$SNR = \frac{P_s}{N_e} \quad (2)$$

P_e = signal-plus-noise energy in the echo filter.
 N_e = noise-only energy in the echo filter.

P_s = signal-only energy in the echo filter.

The procedure for calculating the signal-only energy for individual slices is

$$P_{e,i} = P_{s,i} + N_{e,i} \quad (3)$$

$$SNR = \frac{P_{s,i}}{N_{e,i}} \quad (4)$$

$P_{e,i}$ = signal-plus-noise energy for a slice in the echo filter.

$N_{e,i}$ = noise-only energy for a slice in the echo filter.

$P_{s,i}$ = signal-only energy for a slice in the echo filter.

PROCESSING

Step 1: Calculate the SNR for the whole pulse:

$$SNR = 10 \cdot \text{LOG}_{10}(\text{signal_energy} / \text{echo_noise_energy})$$

Step 2: Calculate the SNR for the center 8 slices:

```
DO i_slice = 1,L1B_slice_dim
  slice_SNR(i_slice) =
    10.*LOG10(signal_energy_slice(i_slice)/echo_noise_energy_slice(i_slice))
END DO
```

Processing Note:

Since the SNR calculation step involves taking the logarithm of a ratio, a special care must be taken to guarantee that the code will not terminate due to a fatal error. Three possible fatal errors are: division by zero, argument of the logarithm is negative, or argument of the logarithm is zero.

The following table summarizes the error processing approach:

| error event | processing | flag associated with error event |
|----------------------|------------|---|
| argument < 1.0e-30 | -300 dB | none |
| argument > 1.0e+30 | 300 dB | none |
| argument is negative | ABS(arg) | The negative sigma0 bit of the sigma0_qual_flag |

CONSTANTS/TABLES

| <u>Name</u> | <u>Description</u> |
|---------------|--|
| L1B_Constants | The run time constants required by the Level 1B Processor. |

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|-------------------------|--------------|--|
| L1B_Constants | | The run time constants required by the Level 1B Processor. |
| L1B_slice_dim | | Maximum slice dimension. |
| echo_noise_energy_slice | DN | Noise-only energy in each slice in echo filter |
| echo_noise_energy | DN | Sum of Noise-only energy in the center 10 slices in the echo filter. |
| signal_energy_slice | DN | Signal-only energy in each slice of the echo filter . |
| signal_energy | DN | Sum of the signal-only energy in the center 10 slices. |

OUTPUT

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|--|
| slice_qual_flag | | Bit flags which indicate the quality of the data for each slice. |
| sigma0_qual_flag | | Bit flag which indicates the quality of the data for a pulse. |
| slice_SNR | dB | The ratio of signal to noise for a slice. |
| SNR | dB | The ratio of signal to noise for a whole pulse. |

REFERENCES

- [1] NASA/JPL Scatterometry Processing Algorithm and Analysis Group, "Science Algorithm Specification for SeaWinds", JPL, October 20, 1996.

SeaWinds Algorithm Specification

TITLE: Compute the Radar Backscattering Coefficient
SUBMODULE: Energy Detection
MODULE: SeaWinds Sigma0 and Kp Algorithm
CODE: L1B.3.4.5
VERSION: 2.0
DATE: 10/31/99
AUTHOR: Kyung Pak and Angela Zhang
SUBROUTINE: Est_Pr_Pt_Ratio
LANGUAGE: FORTRAN
HERITAGE: Est_Ratio_Echo_Trans_Energy (L1B.3.2.4)

L1B.3.4.5 Est_Pr_Pt_Ratio

PURPOSE

This subroutine calculates the ratio of the signal-only energy (Pr) to the transmitted energy (Pt), Pr_Pt_ratio. Instead of calculating Pr and Pt individually, this algorithm evaluates the ratio using the echo return energy at the output of the receiver and a normalization factor P_{cal} .

BACKGROUND

The conventional way to calculate sigma0 from the radar equation requires a separate evaluation of the transmit energy and the receiver energy. The SeaWinds design applies a loop-back calibration scheme. The unique feature of this calibration technique provides a direct way to evaluate Pr / Pt, rather than to evaluate the transmit energy and the receiver energy separately.

The value of this ratio can be calculated from the detected echo energy using the following equation:

$$\frac{P_r}{P_t} = \frac{P_s}{L_{sys} P_{s_cal}} \quad (1)$$

$$L_{sys} = \left[\frac{L_{bias} L_{23}}{L_{13} L_{21}} \frac{L_{cal}}{L_{op}} \right] \quad (2)$$

where

Loss Factors:

L_{13} = Receiver path insertion loss.

| | |
|------------------|--|
| L_{21} | = Transmitter path insertion loss. |
| L_{23} | = Calibration path insertion loss. |
| L_{cal}/L_{op} | = Calibration path attenuator loss. |
| L_{bias} | = Bias between the modeled and measured loss factor. |
| P_{s_cal} | = Normalizing factor derived from the calibration frames, which is obtained in the algorithm Get_Cal_Data.F (L1B.3.3.1). |

The signal-only energy P_s is estimated in Est_Signal_Energy (L1B.3.4.3). Both of these are passed via subroutine arguments to this algorithm.

PROCESSING

Step 1: Calculate the Pr_Pt_ratio for the whole pulse:

$$\text{Pr_Pt_ratio} = 10. * \text{LOG10}(\text{signal_energy} / (\text{L_sys} * \text{signal_energy_cal}))$$

Step 2: Calculate the Pr_Pt_ratio for the center 8 slices:

```
DO i_slice = 1,L1B_slice_dim
  slice_Pr_Pt_ratio(i_slice)
  .
  = 10. * LOG10 (signal_energy_slice(i_slice) / (L_sys*signal_energy_cal) )
END DO
```

Processing Note:

Since the Pr_Pt_ratio calculation step involves taking the logarithm of a ratio, a special care must be taken to guarantee that the code will not terminate due to a fatal error. Three possible fatal errors are: division by zero, argument of the logarithm is negative, or argument of the logarithm is zero.

The following table summarizes the error processing approach:

| error event | processing | flag associated with error event |
|----------------------|------------|---|
| argument < 1.0e-30 | -300 dB | none |
| argument > 1.0e+30 | 300 dB | none |
| argument is negative | ABS(arg) | The negative sigma0 bit of the sigma0_qual_flag |

CONSTANTS/TABLES

| <u>Name</u> | <u>Description</u> |
|---------------|--|
| L1B_Constants | The run time constants required by the Level 1B Processor. |

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|--|
| L1B_slice_dim | | Maximum slice dimension. |
| signal_energy_slice | DN | Signal-only energy in each slice of the echo filter . |
| signal_energy | DN | Sum of the signal-only energy in the center 10 slices. |
| L_sys | dB | The SES calibration loss factor. |
| Signal_energy_cal_A | DN | Calibration energy of beam A pulse. |
| Signal_energy_cal_B | DN | Calibration energy of beam B pulse. |

OUTPUT

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|--|
| slice_Pr_Pt_ratio | dB | The ratio of the echo return power to the transmit power for each slice. |
| Pr_Pt_ratio | dB | The ratio of the echo return power to the transmit power. |

REFERENCE

- [1] NASA/JPL Scatterometry Processing Algorithm and Analysis Group, “Science Algorithm Specification for SeaWinds”, JPL, October 20, 1996.

SeaWinds Algorithm Specification

TITLE: Compute the Radar Backscattering Coefficient
SUBMODULE: Sigma0 and Variance Estimation
MODULE: SeaWinds Sigma0 and Kp Algorithm
CODE: L1B.3.5.1
VERSION: 2.0
DATE: 10/31/99
AUTHOR: Kyung Pak, Vincent Hsiao, and Angela Zhang
SUBROUTINE: Calculate_X_Factor.F
LANGUAGE: FORTRAN
HERITAGE: Calculate_Correction_Factor (L1B.3.3.1)

L1B.3.5.1 Calculate_X_Factor

PURPOSE

This subroutine determines the correction factor (X factor) derived from the true, integrated, antenna and digital filter gain patterns. This algorithm assumes that the X factor values were not precalculated by earlier algorithms such as Compute_X_Loc (L1B.2.4.6). In the event the geometry part of the X factor values were calculated by Compute_X_Loc, this subroutine only combines the calibration part of the X factor with that of the geometry part of the X factor.

BACKGROUND

A backscattering coefficient, sigma0, and a signal-only energy measurement, Ps, is related by the radar equation:

$$P_s = \frac{(P_t G_r) \lambda^2 G_p^2}{(4\pi)^3 L_{sys}} \left(\frac{1}{L_a^2 L_w^2} \right) \iint d^2 \rho \sigma_0(x, y) \frac{F(f_b(\rho))}{R^4(\rho)} \quad (1)$$

where

λ = Wavelength of the transmitted pulse.
 G_p = Antenna peak gain.
 P_s = Signal-only energy for an entire footprint in the echo channel.
 $F(f_b(r))$ = Digital filter response of an entire footprint. The antenna gain pattern is also absorbed in this term.
 $P_t G_r$ = Transmitted power-receiver gain product.

| | |
|-----------------|--|
| $\sigma_0(x,y)$ | = Scattering coefficient at position (x,y) on surface. |
| $R(r)$ | = Slant range to a point (x,y) on surface. |
| L_w | = One way platform waveguide loss (0.21 dB inner beam, 0.24 dB outer beam). |
| L_a | = Atmospheric loss. |

and

$$L_{sys} = \left[\frac{L_{bias} L_{13} L_{21}}{L_{23}} \frac{L_{op}}{L_{cal}} \right] \quad (2)$$

where

| | |
|------------|--|
| L_{bias} | = Bias between the modeled and measured loss factor (0.0 dB for receiver A and 0.08 dB for receiver B) |
|------------|--|

and the loss factor terms are

| | |
|------------------|-------------------------------------|
| L_{13} | = Receiver path insertion loss. |
| L_{21} | = Transmitter path insertion loss. |
| L_{23} | = Calibration path insertion loss. |
| L_{cal}/L_{op} | = Calibration path attenuator loss. |

Under the assumption that s_0 is constant over the footprint, the radar equation can be re-arranged and simplified in terms of the X factor.

$$\sigma_0 = \frac{P_s}{X} \quad (3)$$

The X factor is a normalization factor which can be separated into two parts $X = X_{cal} X_{int}$

where

$$X_{cal} \equiv \frac{\lambda^3}{(4\pi)^3} G_p^2 \left(\frac{1}{L_a^2 L_w^2} \right) \left[\frac{L_{bias} L_{23}}{L_{13} L_{21}} \frac{L_{cal}}{L_{op}} \right] P_{s_cal} \quad (4)$$

$$X_{int} \equiv \iint d^2 f \frac{F(f_b(f))}{R^4(f)} \quad (5)$$

and

| | |
|--------------|--|
| P_{s_cal} | = Signal-only energy of the loop-back calibration pulse. |
|--------------|--|

The signal-only energy P_s is estimated in Est_Signal_Energy (L1B.3.4.3) and X_{cal} is calculated in Est_Calibration_X (L1B.3.3.2). Both of these are passed via subroutine arguments to this algorithm. The individual slice and “egg” X_{int} terms are calculated in this subroutine and combined with X_{cal} to give the total X. The total X are passed on to the subsequent algorithms.

The calculation of X_{int} requires integrating the antenna pattern and digital signal processor responses over the footprint. The computation of the X_{int} for each measurement is CPU time consuming and is more than what the ground processing can handle. In QuikSCAT/SeaWinds, X_{int} is pre-computed and tabularized to significantly reduce the computation time. The X_{int} table is generated in two steps. First, a nominal X_{int} table is generated based on a nominal spacecraft orbit. Perfect Doppler/range tracking algorithm and beam pointing are assumed. This table is called the nominal X_{int} table, X_{nom} . However, the actual orbit, Doppler/range tracking, and beam pointing are different than the nominal orbit, ideal Doppler/range tracking, and perfect beam pointing. Therefore, in the second step the X_{int} values are generated for a range of possible orbits, Doppler/range tracking errors, and attitude errors. Then the differences in the X_{int} and X_{nom} tables as a function of baseband frequency error (or baseband frequency shift) are fitted with a 3rd order polynomial (see the cell location algorithm section for details). The coefficients of this polynomial fit are stored in a X_{coef} table. The X_{coef} table, when used in conjunction with the X_{nom} table, compensates for the difference in the X_{nom} and the actual X_{int} values.

The usage of the X_{nom} and X_{coef} is as follows

$$X_{int} = X_{nom} + A\Delta f + B\Delta f^2 + C\Delta f^3 \quad (6)$$

where the A, B, and C coefficients are the entries in the X_{coef} table. The Δf term is the baseband frequency shift between the nominal geometry and the actual geometry. It is calculated by the following equation

$$\Delta f = -f_c^n - f_{ref}^n - f_s + f_{s'} + f_{proc} - \mu(D_{ref}^n - D_c^n) + f_{topo} \quad (7)$$

where

- f_c^n = Commanded Doppler frequency (Hz).
- f_{ref}^n = Reference vector Doppler frequency (Hz).
- f_s = Modulation chirp start frequency (Hz).
- $f_{s'}$ = Modulation de-chirp start frequency (Hz).
- f_{proc} = -1.06 Hz.
- μ = Frequency modulation chirp rate (Hz/sec).
- D_{ref}^n = Reference vector range delay (sec).
- D_c^n = Commanded range delay (sec).
- f_{topo} = Topographic frequency shift correction (Hz).

The topographic correction algorithm is

$$f_{topo} = S * h \quad (8)$$

where

S = Pre-calculated frequency correction table due to topography (Hz/m).
h = Earth topography map (m).

The baseband frequency shift term without the f_{topo} term is calculated in the Compute_Cell_Geometry (L1B.2.4.0) module and passed to this algorithm.

PROCESSING

Step 1: Calculate the indexes necessary to read the X_{nom} , X_{int} , and S tables and the topo map.
Determine orbit_step and angle.

```
orbit_step = num_orbit_steps * (orbit_time, orbit_period) / rev_orbit_period
int_orbit_step = INT(orbit_step)
int_orbit_step = int_orbit_step + 1
next_orbit_step = int_orbit_step + 1

angle = num_azimuths*(antenna_azimuth)/360.
int_angle = INT(angle)
int_angle = int_angle + 1
next_angle = int_angle + 1
```

Step 2: Determine the bilinear interpolation coefficients a, b, c, and d for the tables.

```
a = orbit_step - int_orbit_step
b = 1. - a
c = angle - int_angle
d = 1.- c
```

Step 3: Interpolate the S table for topography correction and add the term to the total frequency shift. Height is in meters and S is in bin numbers (normalized frequency)/m.

```
s_factor = S_Table(beam_index,int_angle,int_orbit_step,mode_index)*b*d
.      + S_Table(beam_index,next_angle,int_orbit_step,mode_index)*b*c
.      + S_Table(beam_index,int_angle,next_orbit_step,mode_index)*a*d
.      + S_Table(beam_index,next_angle,next_orbit_step,mode_index)*a*c

i_lon = MOD( NINT(cell_lon / 0.25),1440 ) + 1
```

```

i_lat = NINT( (cell_lat + 90.)/0.25 ) + 1
height = MAX( 0.0,Topo_Height(i_lon,i_lat) )

```

```

frequency_shift = frequency_shift + s_factor*height

```

Step 3: Interpolate the X_{nom} table for the individual slices.

```

DO i_slice=1,num_slices_only
  X_factor(i_slice) =
.      (X_nom(i_slice,beam_index,int_angle,int_orbit_step,mode_index))*b*d
.      + (X_nom(i_slice,beam_index,next_angle,int_orbit_step,mode_index))*b*c
.      + (X_nom(i_slice,beam_index,int_angle,next_orbit_step,mode_index))*a*d
.      + (X_nom(i_slice,beam_index,next_angle,next_orbit_step,mode_index))*a*c
  DO i_coef = 1,num_X_coeffs
    X_slice_coef(i_coef,i_slice) =
.      X_coef(i_coef,i_slice,beam_index,int_angle,int_orbit_step,mode_index)*b*d
.      + X_coef(i_coef,i_slice,beam_index,next_angle,int_orbit_step,mode_index)*b*c
.      + X_coef(i_coef,i_slice,beam_index,int_angle,next_orbit_step,mode_index)*a*d
.      + X_coef(i_coef,i_slice,beam_index,next_angle,next_orbit_step,mode_index)*a*c
  END DO
END DO

```

Step 4: Calculate the total X_{int} for the individual slices using equation (6).

```

DO i_slice = 1,num_slices_only
  X_factor(i_slice) = (10.**((X_factor(i_slice)
.      + X_slice_coef(1,i_slice)
.      + X_slice_coef(2,i_slice)*frequency_shift
.      + X_slice_coef(3,i_slice)*frequency_shift**2
.      + X_slice_coef(4,i_slice)*frequency_shift**3)/10.))*X_cal
END DO

```

Step 5: Calculate the total X_{int} for the eggs following the same procedure as in slices.

```

X_egg =
.      X_table(egg_index,beam_index,
.      int_angle,int_orbit_step,mode_index)*b*d
.      + X_table(egg_index,beam_index,
.      next_angle,int_orbit_step,mode_index)*b*c
.      + X_table(egg_index,beam_index,
.      int_angle,next_orbit_step,mode_index)*a*d
.      + X_table(egg_index,beam_index,next_angle,
.      next_orbit_step,mode_index)*a*c

```

```

DO i_coef = 1,num_X_coeffs
  X_factor_egg_coeff(i_coef) =
.      X_coeffs(i_coef,egg_index,beam_index,
.      int_angle,int_orbit_step,mode_index)*b*d
.      + X_coeffs(i_coef,egg_index,beam_index,
.      next_angle,int_orbit_step,mode_index)*b*c
.      + X_coeffs(i_coef,egg_index,beam_index,
.      int_angle,next_orbit_step,mode_index)*a*d
.      + X_coeffs(i_coef,egg_index,beam_index,
.      next_angle,next_orbit_step,mode_index)*a*c
END DO

X_factor_egg = (10.**((X_egg + X_egg_coef(1)
+ X_egg_coef(2)*frequency_shift
+ X_egg_coef(3)*frequency_shift**2
+X_egg_coef(4)*frequency_shift**3)/10.))*X_cal

```

Step 6: Convert X to dB scale.

CONSTANTS/TABLES

| <u>Name</u> | <u>Description</u> |
|---------------|--|
| L1B_Constants | The run time constants required by the Level 1B Processor. |

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|---|
| L1B_slice_dim | | Maximum slice dimension. |
| mode_index | | An identifier indicating the current effective gate width (1-8). |
| beam_index | | Inner/outer beam id (1/2). |
| orbit_time | counts | The current orbit time. |
| rev_orbit_period | sec | The time required for the spacecraft to complete one complete orbit |
| antenna_azimuth | deg | Antenna azimuth angle. |
| frequency_shift | Hz | The shift in the baseband frequency of a pulse due to errors in the Doppler Binning Table and the spacecraft attitude measurement. |
| X_cal | dB | The beam A/B radiometric calibration component of the X-factor. |
| bb_freq_off_corr | | A correction to the calculated change in baseband frequency. This correction reflects biases in the baseband frequency which are due to perturbations in the spacecraft attitude and the antenna look angles. |

| | | |
|-----------------|-----------|---|
| X_Factor_coeffs | | The nominal X table correction factors for individual slices. The 9th slice entry is for the whole egg (spot). |
| X_Factor_Table | $(1/m)^2$ | A table containing the geometric factor in the radar equation that is based on the calculation of an integral. |
| Topo_Height | m | Earth topography map table. |
| S_Table | 1/m | A correction coefficient table used to compensate a baseband frequency error. |

OUTPUT

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|--|
| X_factor | dB | The complete conversion factor from scatterometer received signal energy to sigma0 value for each slice. |

REFERENCES

- [1] NASA/JPL Scatterometry Processing Algorithm and Analysis Group, "Science Algorithm Specification for SeaWinds", JPL, October 20, 1996.

SeaWinds Algorithm Specification

TITLE: Compute the Radar Backscattering Coefficient
SUBMODULE: Sigma0 and Variance Estimation
MODULE: SeaWinds Sigma0 and Kp Algorithm
CODE: L1B.3.5.2
VERSION: 2.0
DATE: 10/31/99
AUTHOR: Kyung Pak and Angela Zhang
SUBROUTINE: Calculate_Sigma0.F
LANGUAGE: FORTRAN
HERITAGE: Calculate_Sigma0 (SeaWinds ATB L1B.3.4.1)

L1B.3.5.2 Calculate_Sigma0

PURPOSE

This subroutine calculates the radar backscattering coefficient, sigma0.

BACKGROUND

The radar backscattering coefficient, σ_0 , and the signal-only energy measurement, P_s , are related by the radar equation. In this section, only a brief description is outlined. For a complete description, refer to the algorithm Calculate_X_Factor.F (L1B.3.5.1). Under the assumption that σ_0 is constant over the footprint, the radar equation can be re-arranged in terms of the X-factor.

$$\sigma_0 = \frac{P_s}{X} \quad (1)$$

The X-factor is a normalization factor which can be separated into two parts $X = X_{cal} X_{int}$. X_{cal} is the instrument calibration measurement calculated in Est_Calibration_X (L1B.3.3.2) and passed to Calculate_X_Factor (L1B.3.5.1). In Calculate_X_Factor the X_{int} term is calculated and combined with X_{cal} . The combined X factor is passed to this subroutine. The signal-only energy P_s is estimated in Est_Signal_Energy (L1B.3.4.3) and passed via subroutine arguments to this algorithm.

PROCESSING

Step 1: Calculate the sigma0 for the individual slices by first checking to see whether the individual cell location algorithm has converged. For the located slices proceed with the sigma0 calculation by:

- a) calculate the slice_sigma0 values using equation (3)
- b) check if slice_sigma0 value is negative. Then clear the negative sigma0 bit flag of the slice_qual_flag for positive valued sigma0s.

DO i_slice = 1, L1B_slice_dim

```

IF (slice_qual .EQ. slice_located) THEN
    slice_sigma0(i_slice) = signal_energy_slice(i_slice)/ (X_factor(i_slice))

    IF(slice_sigma0(i_slice) .GE. 0.) THEN
        slice_qual_flag = My_SWS_Set_Flag
        (slice_qual_flag,slice_sigma0_sign_flag,slice_sigma0_not_negative)
    ELSE
        slice_sigma0(i_slice) = - slice_sigma0(i_slice)
    END IF
    slice_sigma0(i_slice) = 10.* LOG10 (slice_sigma0(i_slice))
ELSE
    slice_sigma0(i_slice) = 0.
END IF

```

END DO

Step 2: Calculate the sigma0 for the whole pulse (egg) as in Step 1:

- a) calculate the slice_sigma0 values using equation (3)
- b) check if slice_sigma0 value is negative. Clear the negative sigma0 bit flag of the sigma0_qual_flag for positive valued sigma0s.

sigma0 = signal_energy/X_factor_egg

```

IF(sigma0 .GE. 0.0) THEN
    sigma0_qual_flag =My_SWS_Set_Flag
    (sigma0_qual_flag,SIGMA0_SIGN_FLAG,SIGMA0_NOT_NEGATIVE)
ELSE
    sigma0 = - sigma0
END IF

```

X_factor_egg = 10.*log10(X_factor_egg)

sigma0 = 10.*log10(sigma0)

Step 3: Set the usability flag by checking the range of sigma0 and instrument temperature values.

```

IF ((sigma0 .LT. MAX_SIGMA0) .AND. (sigma0 .GT. MIN_SIGMA0) ) THEN

```

```

        sigma0_qual_flag = My_SWS_Set_Flag
        (sigma0_qual_flag,SIGMA0_RANGE,SIGMA0_RANGE_ACCEPTABLE)

END IF

IF (sigma0_qual .EQ. TEMP_RANGE_IN_RANGE ) THEN

    sigma0_qual_flag = My_SWS_Set_Flag
    (sigma0_qual_flag,SIGMA0_USABLE_FLAG,SIGMA0_USABLE)

END IF    ! temp. check

END IF    ! sigma0 range check

```

Processing Note:

Since the sigma0 is expressed in dB units, care must be taken to guarantee that the code will not terminate due to a fatal error. Three possible fatal errors are: argument of the logarithm is infinite, negative, or zero.

The following table summarizes the error processing approach:

| error event | solution | flag associated with error event |
|----------------------|----------|---|
| argument < 1.0e-30 | -300 dB | none |
| argument > 1.0e+30 | 300 dB | none |
| argument is negative | ABS(arg) | The negative sigma0 bit of the sigma0_qual_flag |

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|---|
| L1B_slice_dim | | Maximum slice dimension. |
| signal_energy_slice | DN | Signal-only energy in each slice of the echo filter . |
| signal_energy | DN | Sum of the signal-only energy in the center 10 slices. |
| x_factor | dB | The complete conversion factor from scatterometer energy to sigma0 value for each slice of a sigma0 cell. |
| X_factor_egg | dB | The X-factor for a pulse. |
| slice_qual_flag | | Bit flags which indicate the quality of the data for each slice. |
| sigma0_qual_flag | | Bit flag which indicate the quality of the data for a pulse. |

OUTPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|-----------------------------------|
| slice_sigma0 | dB | Radar backscatter for each slice. |
| sigma0 | dB | Radar backscatter for a pulse. |

REFERENCE

- [1] NASA/JPL Scatterometry Processing Algorithm and Analysis Group, “Science Algorithm Specification for SeaWinds”, JPL, October 20, 1996.

SeaWinds Algorithm Specification

TITLE: Compute the Radar Backscattering Coefficient
SUBMODULE: Sigma0 and Variance Estimation
MODULE: SeaWinds Sigma0 and Kp Algorithm
CODE: L1B.3.5.3
VERSION: 2.0
DATE: 10/31/99
AUTHOR: Kyung Pak and Angela Zhang
SUBROUTINE: Calculate_Kpc_Coeff.F
LANGUAGE: FORTRAN
HERITAGE: None

L1B.3.5.3 Calculate_Kpc_Coeff

PURPOSE

Compute Kpc coefficients which are used in level 2B processor to estimate the weighting factors for wind retrieval algorithm.

BACKGROUND

When a scatterometer measures the backscattering coefficient, sigma0, of the ocean surface, it does so with imperfect precision. Due to the instrument thermal noise and radar signal fading effects, the estimates of sigma0 will be noisy. In scatterometry, measurement variations of this type are referred to as “communication noise”. A parameter commonly used to indicate the magnitude of the communication noise is Kpc, which is defined as the normalized standard deviation of the echo return energy:

$$kpc = \frac{\{Var[P_s]\}^2}{P_s} \quad (1)$$

where P_s is the expected echo return energy and $Var[P_s]$ is the variance of the echo return energy.

Kpc is a function of the instrument signal processing parameters and the return signal-only to noise-only energy ratio (SNR). It can be written as:

$$kpc^2 = kpc_A + \frac{kpc_B}{SNR} + \frac{kpc_C}{SNR^2} \quad (2)$$

Kpc_A, kpc_B, and kpc_C are known as the Kpc coefficients. They are functions of the instrument antenna pattern, illumination geometry, modulation format, transmit pulse modulation, and receiver filter characteristics. There are two models to estimate the Kpc coefficients: analog and digital.

Analog Kpc Model

The analog Kpc coefficient derivation is based on an analog signal processing system, as the one employed in Seasat scatterometer [2]. This approach assumes that the bandwidth of each individual slices are narrow so that the power spectral density is flat over the slice bandwidth. Then it can be shown that the Kpc_A, Kpc_B, and kpc_C can be well-approximated by the following set of analog expressions

$$kpc_A = \frac{1}{B_s T_p} \quad (3)$$

$$kpc_B = \frac{2}{B_s T_g} \quad (4)$$

$$kpc_C = \frac{1}{B_s T_g} \left(1 + \frac{B_s}{B_n} \right) \quad (5)$$

where,

B_s = Bandwidth of the individual slices.

B_n = Bandwidth of the noise filter (1MHz nominal).

T_p = Transmit pulse width.

T_g = Range gate width.

The egg Kpc formula results from a similar derivation and approximations

$$kpc_A = \frac{1}{B_{3dB} T_p} \quad (6)$$

$$kpc_B = \frac{2}{B_{egg} T_g} \quad (7)$$

$$kpc_C = \frac{1}{B_{egg} T_g} \left(1 + \frac{B_{egg}}{B_n} \right) \quad (8)$$

where

B_{3dB} = The 3dB bandwidth of the egg .

B_{egg} = The sum of the individual slice bandwidths comprising an egg (nominally there are 10 slices in an egg).

Digital Kpc Model

In the digital Kpc formulation, the shape of the power spectral density and the digital filter effects are carefully modeled. This results in nearly constant Kpc_B and Kpc_C values which can be well-approximated by equations (4)-(5) and (7)-(8). The digital Kpc_A varies about 15 % as a function of antenna azimuth angle. The equation is much more complicated than the analog expression, making pulse-by-pulse calculations impractical. Thus, an approximation which results in a simpler Kpc_A equation or a tabularized Kpc_A values for a quick look up is preferred in the level 1B processing. A simple approximation is to use the analog equations (3) and (6), and accept the maximum of 15% error. More accurate approach is to tabularize the Kpc_A coefficient as a function of orbit position, azimuth angle, beam number, and slice number.

In this subroutine, the analog equation approximation algorithm is implemented as the nominal algorithm since the Kpc_A table is not yet available. The Kpc_A table look-up approach will be the alternate algorithm. The table look up approach is more accurate of the two, and is the preferred method once the Kpc_A table is available.

Kpc_A is computed at discrete scan angles (2 deg.) and orbit positions (10 deg.) to form a table of dimension (180,36). This algorithm retrieves kpc_A for a given scan angle and orbit position by looking up the nearby values in the table and using bilinear interpolation to determine the appropriate value.

The indices for argument of latitude are calculated by:

$$\begin{aligned} i_orb_left &= arg_lat / 10 \\ i_orb_right &= i_orb_left + 1. \end{aligned} \quad (3)$$

The indices of antenna azimuth angle are calculated by:

$$\begin{aligned} i_az_left &= antenna_azimuth / 2 \\ i_az_right &= i_az_left + 1. \end{aligned} \quad (4)$$

The algorithm retrieves the value from the table at the following four locations:

$$\begin{aligned} y1 &= kpc_table(ibeam, i_az_left, i_orb_left) \\ y2 &= kpc_table(ibeam, i_az_right, i_orb_left) \\ y3 &= kpc_table(ibeam, i_az_right, i_orb_right) \\ y4 &= kpc_table(ibeam, i_az_left, i_orb_right). \end{aligned} \quad (5)$$

Then it calculates fractions t and u by

$$\begin{aligned} t &= (antenna_azimuth - 2 * float(i_az_left)) / 2.0 \\ u &= (arg_lat - 10 * float(i_orb_left)) / 10.0 \end{aligned} \quad (6)$$

Finally, it calculates the Kpc_A by

$$Kpc_A = (1-t)*(1-u)*y1 + t*(1-u)*y2 + t*u*y3 + (1-t)*u*y4 \quad (7)$$

The QuikSCAT/SeaWinds instrument is designed to be capable of operating at different resolution modes (effective gate width). Each resolution mode results in a unique set of slice bandwidth and range gate width which can then be used to calculate Kpc_B and Kpc_C. They are essentially functions of resolution mode only. In level 1B processor, the Kpc_B and Kpc_C terms for all resolution modes are pre-calculated and stored in an array for use in the level 2A algorithms.

PROCESSING

Analog Kpc_A Processing

Step 1: Loop through the slices and calculate the Kpc_A using equation (3)

```
DO i_slice = 1,L1B_slice_dim
    slice_qual = My_SWS_Get_Flag(slice_qual_flag,slice_loc_flag)
    IF (slice_qual .EQ. slice_located) THEN
        slice_Kpc_A(i_slice)
            = 1./(tx_pulse_width* L1B_Constants.Bs(mode_index)/10.)
    END IF
END DO
```

Step 2: Calculate egg Kpc_A using equation (6)

```
IF (sigma0_qual .NE. SIGMA0_NOT_USABLE) THEN
    sigma0_Kpc_A
        = 1./ (tx_pulse_width*BANDWIDTH_3DB(beam_index))
END IF
```

Table-driven Kpc_A Processing

Step 1: Orbital indices calculation

compute i_orbit_left and i_orb_right using Eq (3).

Step 2: Antenna azimuth indices calculation

compute i_{az_left} and i_{az_right} using Eq (4)

Step 3 Retrieval of table values

Retrieve kpc_A values at the grid points using Eq (5)

Step 4: Fractional parts calculation

calculate t and u using Eq. (6)

Step 5: Kpc_A calculation

calculate the value of Kpc_A .

CONSTANTS/TABLES

| <u>Name</u> | <u>Description</u> |
|----------------|--|
| L1B_Constants | The run time constants required by the Level 1B Processor. |
| Kpc_Parameters | Coefficients which are used to determine the normalized standard deviation of the σ_0 measurement due to the statistical nature of the received signal, or kpc_A . |

INPUTS

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|------------------------|--------------|--|
| L1B_slice_dim | | Maximum slice dimension. |
| mode_index | | An identifier indicating the current effective gate width (1-8). |
| beam_index | | Inner/outer beam id (1/2). |
| slice_qual_flag | | Bit flags which indicate the quality of the data for each slice. |
| $\sigma_0_qual_flag$ | | Bit flag which indicate the quality of the data for a pulse. |
| range_gate_width | sec | The range gate width for beam A/B. |
| tx_pulse_width | sec | The transmit pulse width. |

OUTPUT

| <u>Variable Name</u> | <u>Units</u> | <u>Description</u> |
|----------------------|--------------|---|
| slice_Kpc_A | | The normalized standard deviation of the slice signal. |
| $\sigma_0_Kpc_A$ | | The normalized standard deviation of the whole echo signal. |

REFERENCES

- [1] NASA/JPL Scatterometry Processing Algorithm and Analysis Group, "Science Algorithm Specification for SeaWinds", JPL, October 20, 1996.

- [2] Fisher, R., "Standard deviation of scatterometer measurements from space," IEEE Transactions on Geoscience and Electronics, Vol. GE-10, No. 2, April 1972.

SeaWinds Scatterometer Brightness Temperature Algorithm

Module L1B.4.0

ALGORITHM SPECIFICATIONS

| | |
|-----------------|----------------|
| AUTHOR: | Barry H. Weiss |
| VERSION: | 1.0 |
| DATE: | May 15, 2001 |

SeaWinds Scatterometer Brightness Temperature Algorithm

MODULE L1B.4.0

I. MODULE OVERVIEW

Physical phenomena associated with precipitation have been found to have an adverse effect on the accuracy of scatterometer measurements. Heavy clouds and light precipitation cause attenuation of the radar signal. Furthermore, moderate to heavy rain backscatters the radar signal more than it attenuates it. Moderate to heavy rain can also roughen the ocean surface such that the backscatter is no longer closely related to the near-surface wind. While deviations of the scatterometer measurements from the wind speed model function may give an indication that the measurements are contaminated by rain (see description of Multi-Dimensional Histogram Rain Flag [MUDH]), radiometer measurements provide an independent observation to assist in this determination. While the scatterometer was not designed as a radiometer, the 1 MHz noise channel provides at least a minimal capability for measuring the Ku band brightness temperature.

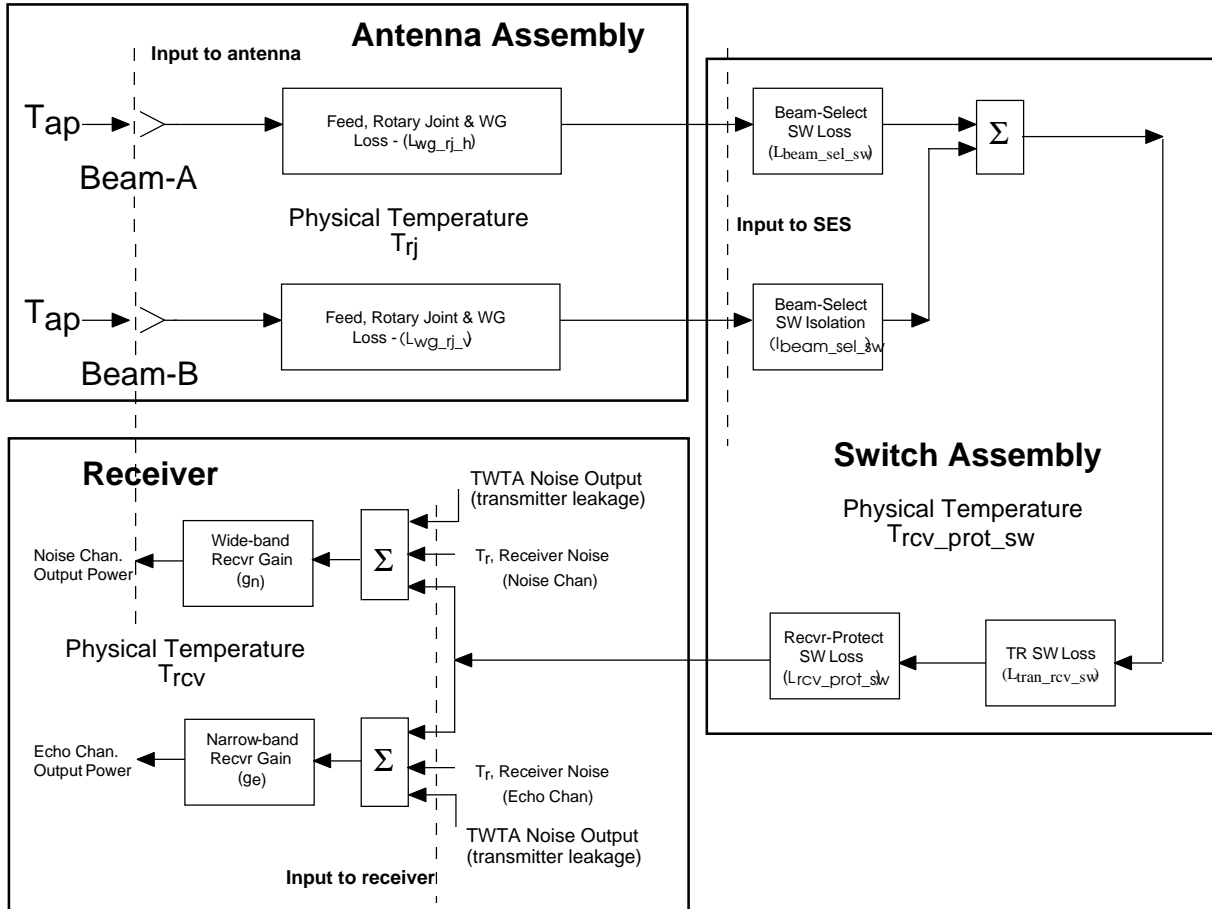
During the QuikSCAT cal/val phase, it became clear that some wind measurements show the effects of precipitation. A means of detecting this contamination was needed so that measurements with a high probability of rain could be flagged. Data users who choose to employ rain flagged data should then acknowledge the possibility that the observations are corrupted.

The same precipitation conditions that generate attenuating atmospheric conditions for scatterometer measurements also generate dramatic increases in the apparent brightness temperatures at microwave frequencies over bodies of water. Thus, apparent brightness temperatures can provide a means to detect the presence of precipitation. Of course, any apparent brightness temperature observations that might be used to detect rainfall need to be both contemporaneous and colocated with the scatterometer data. These narrow time and space requirements led to the idea that the scatterometer measurements, while not ideal, could be used to generate useful apparent brightness temperatures for the purpose of flagging wind measurements for rain contamination.

The algorithm in this module employs the noise channel and echo channel measurements from the scatterometer to produce greybody emission measures at both vertical and horizontal polarization. The algorithm subtracts the echo signal from the corresponding noise channel measurement. Once the energy associated with the scatterometer signal has been removed, the algorithm accounts for other factors that might contribute to the residual noise signal. These include emissions from waveguide losses, leakage from the Travelling Wave Tube Amplifier (TWTAs) and internal receiver noise.

The following block diagram displays the expected instrument loss factors and energy sources that contribute the residual noise from which the brightness temperatures are extracted.

SeaWinds Radiometer Diagram - Chan-A Receiving



II. FUNCTIONAL FLOW DESCRIPTION

The design divides the algorithm into two major segments.

Generate Parameters for Brightness Temperature Calculation

This segment computes those parameters that are characteristic of all observations in a single telemetry frame. Several of these parameters are smoothed over a time series of measurements before algorithmic calculations are applied. Only one sequence of Level 1B Processor code views data over the entire rev, and can thus calculate parameters over a period of time within the rev. This sequence of Level 1B code uses the Calibration Pulse Product, which is generated in the Level 1A Processor, to gain access to data over of the entire time span of the rev. Thus, the Level 1B Processor calls this brightness temperature preparation algorithm within a sector of the code that processes the Calibration Pulse Product data.

Calculate Brightness Temperatures

This segment calculates an individual brightness temperature for each measurement pulse in each telemetry frame. The Level 1B Processor calls this algorithm from the Compute Sigma0 and Kp module.

SeaWinds Algorithm Specification

| | |
|--------------------|--|
| TITLE: | Generate Parameters for Brightness Temperature Calculation |
| SUBMODULE: | |
| MODULE: | Calculate Scatterometer Brightness Temperature |
| CODE: | L1B.4.1.1 |
| VERSION: | 1.0 |
| DATE: | 05/03/01 |
| AUTHOR: | Barry Weiss |
| SUBROUTINE: | Calculate_Tb_Parameters |
| LANGUAGE: | Fortran 90 |
| HERITAGE: | None |

L1B.4.1.1 Calculate_Tb_Parameters

PURPOSE

This algorithm calculates a set of common parameters that will be used to generate apparent brightness temperatures for all of the measurement pulses in a single telemetry frame.

BACKGROUND

This algorithm determines representative noise temperature biases and a representative instrument loss factor for each telemetry frame. The Level 1B algorithm subsequently employs these quantities to estimate that portion of the noise signal that is due to greybody emission from the earth's surface.

The temperature biases and loss factor are based on three spacecraft temperature measurements. These are the receiver temperature T_{rcv} , the rotary joint temperature T_{rj} and the receiver protect switch temperature $T_{rcv_prot_sw}$. The Level 1B Processor reads these data from the Calibration Pulse Product, which is generated by the Level 1A Processor. Before the calculation of the noise bias and loss factor, the algorithm smoothes each of these input temperature measurements over time. The code applies a rolling average of the temperature measurements that were acquired between one minute before and one minute after the current telemetry frame to smooth each of these temperatures. The tabular variable `qscat_rad_temp_dim` stores the number of measurements that normally would span the requisite two minute period. In the current implementation, `qscat_rad_temp_dim` is 78.

Upon completion of the rolling averages, the algorithm computes the temperature biases and the loss factor that are used in the final calculation of the apparent brightness temperature. The loss factor is the receiver protect switch loss $L_{rcv_prot_sw}$. This loss factor calculation (linear power ratio) is based on a second order polynomial fit of the averaged receiver protect switch

temperature.

$$L_{rcv_prot_sw} = A_{rcv_prot_sw2} * T_{rcv_prot_sw}^2 + A_{rcv_prot_sw1} * T_{rcv_prot_sw} + A_{rcv_prot_sw0} \quad (1)$$

The first temperature bias is the receiver radiometric noise temperature T_{rcv_rad} , which is based on the averaged receiver temperature:

$$T_{rcv_rad} = (nf - 1) * T_{noise_figure_reference} \quad (2)$$

where nf, or the noise figure is:

$$nf = 10^{(A_{receiver1} * T_{rcv} + A_{receiver0})} \quad (3)$$

The second bias is the waveguide radiometric noise temperature, which accounts for instrument losses between the antenna and the receiver input. Losses that contribute to the waveguide radiometer bias include those of the feed, the rotary joint and the waveguide; the beam select switch; the transmitter switch; and the receiver protect switch. The means of calculating the waveguide radiometric bias temperature for either polarization is:

$$T_{waveguide_ipol} = L_{rcv_prot_sw} * [(1 - L_{wg_rj_ipol}) * T_{rj} * L_{beam_sel_sw} * L_{tran_rcv_sw} + (1 - L_{beam_sel_sw}) * T_{rcv_prot_sw} * L_{tran_rcv_sw} + (1 - L_{tran_rcv_sw}) * T_{rcv_prot_sw}] + (1 - L_{rcv_prot_sw}) * T_{rcv_prot_sw} \quad (4)$$

The subscript ipol references the polarization that is being processed. Thus, ipol can represent either horizontal or vertical polarization.

Jones[1] accounts for several loss factors in his formulation of the algorithm to calculate apparent brightness temperature based on the excess detected noise in the scatterometer measure. To simplify the form of the ultimate algorithm, he combines the mathematical terms that express these losses. The outcome contains four bias temperatures and loss factors, all of which are attributable to multiple sources. Two of these are calculated once per telemetry frame. Jones names these the Y_bias and the Z_Factor.

The algorithm for the Y_bias factor, that relates to the leakage of the opposite antenna loss bias temperature, is:

$$Y_{ipol} = T_{rcv_prot_sw} * A_{y_factor_ipol} \quad (5)$$

where

$$A_{y_factor_ipol} = (1 - L_{wg_rj_opol}) * L_{beam_sel_sw} \quad (6)$$

The subscript ipol references the polarization that is being processed. Thus, ipol can represent either horizontal or vertical polarization. The subscript opol in the independent

variable $L_{wg_rj_opol}$ infers the opposite of the polarization that is referenced in the dependent variable $A_{y_factor_ipol}$. Thus, to calculate $A_{y_factor_v}$, one would apply the loss factor $L_{wg_rj_h}$. Likewise, to calculate $A_{y_factor_h}$, one would apply the loss factor $L_{wg_rj_v}$.

The algorithm for the Z factor is:

$$Z = L_{tran_rcv_sw} * L_{rcv_prot_sw} \quad (7)$$

The apparent brightness temperature algorithm requires a representative measure of the noise channel calibration energy E_{n_cal} . The load calibration pulse measurement provides the source of the noise channel calibration energy. The algorithm smoothes the load calibration pulse measurement over time. A rolling average over a range of calibrations that are nearest to the time of the current telemetry frame performs this smoothing function. The tabular variable `qscat_rad_noise_dim` stores the number of calibration measurements that contribute to each rolling average calculation. In the current implementation, `qscat_rad_noise_dim` is 120.

INPUTS

The algorithm reads all four of these input parameters from the Calibration Pulse Product that is generated by the Level 1A Processor:

| | |
|---------------------|--|
| T_{rcv} | temperature of the instrument receiver |
| $T_{rcv_prot_sw}$ | temperature of the instrument receiver protect switch |
| T_{rj} | temperature of the instrument rotary joint |
| Cal_{load} | ambient temperature load scatterometer noise calibration |

OUTPUTS

| | |
|-----------------------|--|
| $T_{waveguide_ipol}$ | waveguide radiometric bias temperature. One element applies for vertical polarization, the other applies for horizontal polarization. |
| Y_{ipol} | y-factor. This relates to cross-polarization leakage bias. One element applies for vertical polarization, the other element applies for horizontal polarization. |
| Z | z-factor |
| E_{n_cal} | noise channel calibration energy |
| T_{rcv_rad} | the radiometric noise temperature attributed to the instrument receiver |

CONSTANTS

All of the following constants are stored in the Level 1B Constants Table:

| | | |
|--------------------------------|--|------------|
| $A_{rcv_prot_sw2}$ | second order coefficient to calculate receiver protect switch loss | 1.4413e-6 |
| $A_{rcv_prot_sw1}$ | first order coefficient to calculate receiver protect switch loss | -1.8054e-3 |
| $A_{rcv_prot_sw0}$ | zero order coefficient to calculate receiver protect switch loss | 1.1213 |
| $T_{noise_figure_reference}$ | noise figure reference temperature | 290.0 |
| $A_{receiver1}$ | first order coefficient used to calculate noise figure ratio | 5.333e-4 |

| | | |
|---------------------------------|--|-------------|
| $A_{\text{receiver0}}$ | zero order coefficient used to calculate noise figure ratio | 0.21226 |
| $L_{\text{wg_rj_v}}$ | rotary joint and platform wave guide loss ratio for v-pol | 0.7842 |
| $L_{\text{wg_rj_h}}$ | rotary joint and platform wave guide loss ratio for h-pol | 0.7730 |
| $L_{\text{beam_sel_sw}}$ | beam select switch loss factor | 0.9772 |
| $L_{\text{tran_rcv_sw}}$ | transmit/receive switch loss factor | 0.9772 |
| $A_{\text{y_factor_v}}$ | y-factor coefficient for vertical polarization | 1.790712e-3 |
| $A_{\text{y_factor_h}}$ | y-factor coefficient for horizontal polarization | 1.70236e-3 |
| $I_{\text{beam_sel_sw}}$ | beam select switch isolation ratio | 7.88863e-3 |
| $\text{qscat_rad_temp_dim}$ | number of temperature measurements applied to each rolling average calculation | 78 |
| $\text{qscat_rad_noise_dim}$ | number of noise measurements applied to each rolling average calculation | 120 |

PROCESSING

Step 1: Calculate a rolling average of the receiver temperature, the receiver protect switch temperature, and the rotary joint temperature over the $\text{qscat_rad_temp_dim}$ telemetry frames which are nearest in time to the current telemetry frame.

Step 2: Use equation (1) to calculate the receiver protect switch loss.

Step 3: Use equations (2) and (3) to calculate the instrument receiver radiometric temperature.

Step 4: Use equation (7) to apply the rolling averaged receiver protect temperature to calculate the Z factor.

Step 5: Use equations (5) and (6) to calculate the Y factor for each polarization.

Step 6: Use equation (4) to calculate the waveguide radiometric bias temperature for each polarization.

Step 7: Calculate $E_{\text{n_cal}}$, a rolling average of the noise channel load calibration Cal_{load} over the $\text{qscat_rad_noise_dim}$ calibration measures that are nearest in time to the current telemetry frame.

REFERENCE

- [1] Jones, Linwood, Second Generation QuikSCAT Radiometer Apparent Brightness Temperature Algorithm Rev-B, University of Central Florida Remote Sensing Laboratory, Orlando, Florida, March 16, 2000.

SeaWinds Algorithm Specification

| | |
|--------------------|---|
| TITLE: | Calculate Scatterometer Based Brightness Temperatures |
| SUBMODULE: | |
| MODULE: | Calculate Scatterometer Brightness Temperature |
| CODE: | L1B.4.1.2 |
| VERSION: | 1.0 |
| DATE: | 04/23/01 |
| AUTHOR: | Barry Weiss |
| SUBROUTINE: | Calculate_Brightness_Temperature |
| LANGUAGE: | Fortran 90 |
| HERITAGE: | None |

L1B.4.1.2 Calculate_Brightness_Temperature

PURPOSE

This algorithm calculates an apparent brightness temperature value for each measurement acquired by the SeaWinds scatterometer. These brightness temperatures are subsequently averaged with nearby measurements of the same polarization and then used in the Level 2B Processor to detect the likelihood of precipitation. Relative to a radiometer, the brightness temperatures that are extracted from the scatterometer measures are not very accurate. However, the presence of rain over open water surfaces tends to dramatically increase the apparent brightness temperature. Thus, these measurements remain effective detectors of precipitation that are known to have adverse effects on scatterometer measurements.

BACKGROUND

The Level 1B Processor calls Calculate_Brightness_Temperature from the driver subroutine Calculate_Sigma0_and_Kp. Calculate_Sigma0_and_Kp executes a loop over all of the scatterometer pulses in the current telemetry frame. The same code removes any calibration pulses from the processing sequence. Thus, each call to this algorithm processes one measurement scatterometer pulse.

Most of the input parameters to this subroutine are calculated in Calculate_Tb_Parameters. The Level 1B Processor determines the value of two other prerequisite parameters before the call to Calculate_Brightness_Temperature. One of these prerequisites is the bandwidth ratio a , which is calculated in Process_Calibration_Data. The second is the sum of the echo energy for all twelve slices in the current scatterometer pulse, or E_{echo} . E_{echo} is calculated in Est_Noise_Energy.

The algorithm in Calculate_Brightness_Temperature determines the apparent brightness temperature in three steps. First, the algorithm calculates a weighted difference between the

noise channel output energy and echo channel output energy. Jones [1] refers to this value as the excess noise.

$$N_{\text{excess}} = (E_{\text{noise}} - \beta_{\text{rad}} * E_{\text{echo}}) * (\alpha - 1) / (\alpha - (1 + \epsilon))$$

where

α is the noise channel to echo channel bandwidth ratio

β_{rad} is the ratio of the mean noise channel gain to the mean echo channel gain

ϵ is the noise energy ratio between modulation on and modulation off operational modes

The algorithm uses the excess noise to calculate an effective total radiometric temperature that accounts for all sources of the excess noise.

$$T_{\text{eff_ipol}} = [N_{\text{excess}} * (T_{\text{rcv_prot_sw}} + T_{\text{rcv_rad}}) / C_{\text{eff_load_cal}}] / [E_{\text{n-cal}} * (1 - 1/\alpha)]$$

where

$C_{\text{eff_load_cal}}$ is the effective load calibration factor

$T_{\text{eff_ipol}}$ is the effective total radiometric temperature for the polarization of the corresponding measurement pulse.

The algorithm now calculates the apparent brightness temperature. This method removes three precalculated contributors from the effective total radiometric temperature, and then adjusts the outcome with a series of instrument noise biases and loss factors. Note that the algorithm also includes first order and zeroth order coefficients that can be used to adjust these results. In the QuikSCAT implementation, the first order coefficient for both the vertical and the horizontal polarization is one. The zeroth order coefficient for both the vertical and the horizontal polarization is zero.

$$T_{\text{app_ipol}} = C_{\text{slope_ipol}} * \{ [(T_{\text{eff_ipol}} - T_{\text{waveguide_ipol}} - T_x - T_{\text{rcv_rad}}) / Z - D_{\text{ipol}} - Y_{\text{ipol}}] / X_{\text{ipol}} \} + C_{\text{offset_ipol}}$$

where

T_x is the transmitter leakage bias. This value is provided in the L1B Constants Table.

D_{ipol} is the D loss factor. The Level 1B Constants Table contains two D factor values.

One applies for vertical polarization, the other for horizontal polarization.

X_{ipol} is the X loss factor. The Level 1B Constants Table contains two X factor values.

One applies for vertical polarization, the other for horizontal polarization.

$C_{\text{slope_ipol}}$, $C_{\text{offset_ipol}}$ - calibrates the outcome of the brightness temperature calculation to correlate with measurements acquired using the Tropical Rainfall Measuring Mission (TRMM) Microwave Imager (TMI).

INPUTS

The algorithm reads the following input parameter directly from the Level 1A Product:

| | |
|--------------------|--|
| ipol | index that identifies the polarization of the beam being processed |
| E _{noise} | the noise channel energy for one observation, contains echo plus noise |

Subroutine Calculate_Tb_Parameters calculates each of the following input parameters:

| | |
|-----------------------------|--|
| T _{waveguide_ipol} | waveguide radiometric bias temperature. The input is specific to the polarization of the active scatterometer beam. Thus, ipol can represent either vertical or horizontal polarization. |
| Y _{ipol} | y-factor noise bias. One element applies for vertical polarization, the other element applies for horizontal polarization. |
| Z | z-factor |
| E _{n-cal} | noise channel calibration energy |
| T _{rcv_rad} | the radiometric noise temperature attributed to the instrument receiver |

Subroutine Process_Calibration_Data determines the following value:

| | |
|----------|---|
| α | the bandwidth ratio (noise channel to echo channel) |
|----------|---|

Subroutine Est_Noise_Energy determines the following value:

| | |
|-------------------|---|
| E _{echo} | the sum of the twelve slice echo energies for a scatterometer observation |
|-------------------|---|

OUTPUTS

| | |
|-----------------------|--|
| T _{app_ipol} | The apparent brightness temperature. The output is specific to the polarization of the active scatterometer beam. Thus, ipol can represent either vertical or horizontal polarization. |
|-----------------------|--|

CONSTANTS

All of the following constants are stored in the Level 1B Constants Table:

| | | |
|---------------------------|--|-----------|
| β_{rad} | ratio of the mean noise channel gain to the mean echo channel gain | 2.917427 |
| C _{eff_load_cal} | effective load calibration factor | 0.952 |
| T _x | the transmitter leakage bias | 2.0 |
| X _v | x-factor for vertical polarization | 0.772418 |
| X _h | x-factor for horizontal polarization | 0.761562 |
| D _v | d-factor for vertical polarization | -0.426852 |
| D _h | d-factor for horizontal polarization | 0.433037 |
| C _{slope_v} | correlation slope for vertical polarization | 1.0 |

| | | |
|------------------------|--|-----|
| $C_{\text{slope_h}}$ | correlation slope for horizontal polarization | 1.0 |
| $C_{\text{offset_v}}$ | correlation offset for vertical polarization | 0.0 |
| $C_{\text{offset_h}}$ | correlation offset for horizontal polarization | 0.0 |

PROCESSING

Step 1: Calculates the excess noise for the scatterometer measurement pulse.

Step 2: Calculates the effective total radiometric temperature for the polarization of the current scatterometer pulse.

Step 3: Calculates the apparent brightness temperature for the polarization of the current scatterometer pulse.

REFERENCE

- [1] Jones, Linwood, Second Generation QuikSCAT Radiometer Apparent Brightness Temperature Algorithm Rev-B, University of Central Florida Remote Sensing Laboratory, Orlando, Florida, March 16, 2000.

Track Echo Signal

Module L1B.5.0

ALGORITHM SPECIFICATIONS

| | |
|-----------------|---|
| AUTHORS: | R. Scott Dunbar S. Vincent Hsiao Philip S. Callahan Kyung S. Pak |
| VERSION: | 1.0 |
| DATE: | April 25, 2001 |

Track Echo Signal

MODULE L1B.5.0

I. Module Overview

The SeaWinds high-resolution mode provides return signal power data for 12 "slices" of each received pulse. These slices are resolved in range by transmitting a chirped pulse to the target and Fourier transforming the time history of the return into frequency space. Given the spacecraft orbit and assuming no attitude errors, we can compute the expected frequency at which the return from the antenna peak gain should be seen. Attitude errors introduce small shifts in both range and doppler frequency which are manifested in shifts of the frequency of the peak return.

Wu [1] introduced the idea of tracking the echo return frequency to estimate the spacecraft attitude errors (roll, pitch, yaw). The formulation of the technique was soon after refined by Dunbar [2], showing that the attitude errors could be uniquely determined in all three components, although the yaw dependence was the weakest. Further work by Hsiao has demonstrated that the attitude error can be tracked by this method with relatively few SeaWinds scans. The estimates for pitch and roll obtained by constraining the yaw value were found to be consistently better than the estimates for all three components obtained from an unconstrained analysis.

The echo-tracking technique has been incorporated into the SeaWinds Level 1B algorithm set. It is expected that it will be used for validation of the ADEOS-II horizon-sensor attitude data, and possibly for generating the attitude data actually used in the SeaWinds cell location processing.

This algorithm module consists of three main submodules. Track Echo Signal (L1B.5.1.0) is the main driver that manages the echo-tracking algorithm processing. Manage Echo Track Frames (L1B.5.2.0) deals with the accumulation and buffering of data required for subsequent calculations, including gap handling. Acquire Echo Track Matrix (L1B.5.3.0) computes the components of the attitude determination matrix from spacecraft and cell geometry data. Finally, Calculate Echo Track Attitude (L1B.5.4.0) performs the actual inversion of the echo track matrix and generates the estimated spacecraft roll, pitch, and yaw. Other required calculations related to spacecraft and cell location geometry use algorithms from modules L1B.1 (Spacecraft Location) and L1B.2 (Geometry).

II. Functional Flow Description

The functional flow of this algorithm module is depicted in Figure 1. There are nine subroutines that are specific to the echo-tracking calculations for which the specifications are given in this module. These subroutines are highlighted in the figure. Other geometry algorithms used elsewhere in the Level 1B processing are also used in this algorithm to generate some of the data required by the echo-tracking calculations.

Track_Echo_Signal (L1B.5.1.0)

This is the top-level driver for the echo-tracking algorithm calculations, called by the `Execute_L1B_Algorithms` routine in the Level 1B processor.

Manage_Echo_Track_Frames (L1B.5.2.0)

This routine sets up and controls the buffering of data for the echo-tracking algorithm. For each incoming L1A frame number and time tag, it determines whether the buffer contains sufficient contiguous data to produce an attitude record, whether and where there are gaps in the buffered data, and when the buffer needs to be re-initialized. Buffer initialization is performed by the `Initialize_Echo_Track` (L1B.5.2.1) subroutine. The `Track_Frame_Times` (L1B.5.2.2) subroutine is called later in the echo tracking to keep a running count of the number of frames in the buffer, and the cumulative frame times in the buffer (to generate a mean attitude measurement time tag).

Acquire_Echo_Track_Matrix (L1B.5.3.0)

This routine computes the contributions of the current frame to the matrix elements involved in the later attitude estimation, and maintains the cumulative values of those matrix elements. Prior to this, at convenient points in the general cell geometry computations that are required for echo tracking, the subroutines `Ascertain_Beam_Parameters` (L1B.5.3.1) and `Determine_Echo_Track_Parameters` (L1B.5.3.2) are called. `Ascertain_Beam_Parameters` (L1B.5.3.1) determines the beam (inner/outer) and other key instrument parameters needed for the geometric calculations for each pulse. `Determine_Echo_Track_Parameters` (L1B.5.3.2) computes the orbit radius and the Earth radius at the nadir, which can be computed once per frame.

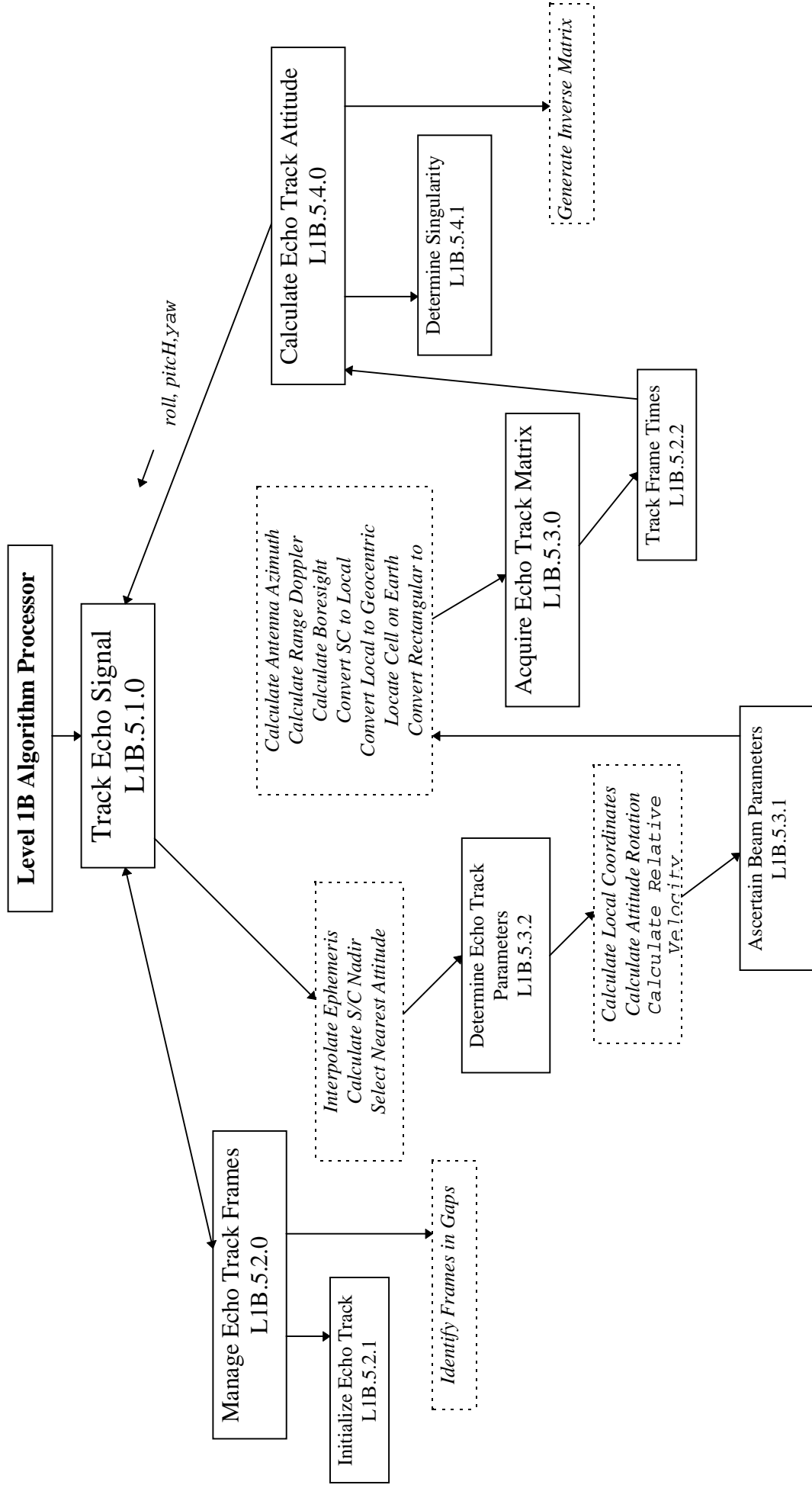
Calculate_Echo_Track_Attitude (L1B.5.4.0)

This submodule performs the final attitude estimation from the accumulated data in the echo-track buffer, producing a finished attitude record. The computation involves the inversion of the least-squares normal matrix for which the elements have been computed in the `Acquire_Echo_Track_Matrix` (L1B.5.3.0) submodule. An invertibility check is performed by the subroutine `Determine Singularity` (L1B.5.4.1) to compute the determinant of the normal matrix before the inversion is attempted.

III.Implementation Assumptions

The Echo Tracking algorithm is the first of the L1B algorithms to be executed from within the L1B processor. It manages a buffer of data required for the eventual attitude determination step, generating an attitude state (roll, pitch, yaw) for each 20 seconds of data. All of the geometric computations are performed relative to the pulse centers, using whatever input attitude data, if any, is available to generate the echo tracking inputs. The output attitude state is then returned to the main L1B algorithm processor for use in the actual cell location and backscatter calculations.

Figure 1. Sea Winds Echo Tracking Algorithm Functional Flow



= Echo Tracking algorithm module

= L1B geometry modules (also used in echo

SeaWinds Algorithm Specification

| | |
|--------------------|---|
| TITLE: | Track Echo Signal |
| SUBMODULE: | |
| MODULE: | Track Echo Signal |
| CODE: | L1B.5.1.1 |
| VERSION: | 1.0 |
| AUTHOR: | R. Scott Dunbar, S. Vincent Hsiao, Philip S. Callahan |
| SUBROUTINE: | Track_Echo_Signal |
| LANGUAGE: | FORTRAN 90 |
| HERITAGE: | None |

L1B.5.1.1 Track_Echo_Signal

PURPOSE

This module is the main driver for the Level 1B Processor echo tracking algorithm. The echo tracking algorithm provides an alternative means to determine the attitude of the spacecraft. The algorithm detects the peak of the echo signal returned to the scatterometer and compares the observed frequency shift of that peak against the expected nominal shift assuming no attitude errors. By accumulating these frequency shift data from several scans, the actual attitude of the spacecraft, relative to a nadir-referenced body coordinate system, can be estimated.

BACKGROUND

The SeaWinds high-resolution mode provides return signal power data for 12 "slices" of a received echo pulse. These slices are resolved in range by transmitting a chirped pulse to the target and Fourier transforming the time history of the return into frequency space. Given the spacecraft orbit and assuming no attitude (roll, pitch, yaw) errors, we can compute the expected frequency at which the return from the antenna peak gain should be seen. Attitude errors introduce small shifts in both range and doppler frequency which are manifested in shifts of the frequency of the peak return. If $\delta\mathbf{R}$ is the change of the slant-range vector to the target due to attitude errors, then the combined doppler- and range-related frequency shift is given in general by:

$$\Delta F = \Delta F_R + \Delta F_D = (2\mu/c) \delta\mathbf{R} \cdot \mathbf{R}_u + (2F/c) d\mathbf{R}_u \cdot \mathbf{V}_r$$

where F is the transmit frequency in Hz, μ is the chirp rate in Hz/sec, c is the speed of light, \mathbf{R} and $\delta\mathbf{R}$ are the slant range vector and its attitude-induced shift, \mathbf{R}_u and $d\mathbf{R}_u$ are their respective unit vectors, and \mathbf{V}_r is the relative velocity of the spacecraft and the target location. The total ΔF is thus a function of the spacecraft location and velocity, the target location, and the attitude of

the spacecraft. Given the spacecraft orbit, the frequency shift can be expressed as a function of the scan azimuth θ in the form:

$$\Delta F = A \cos \theta + B \sin \theta + C$$

where the coefficients A, B, and C are in turn functions of the orbit (spacecraft location and relative velocity), instrument look angle, and attitude.

The orbit and look angle dependencies can be computed separately from the usual cell location calculations as coefficients of the attitude components, giving three linear equations for A, B, and C with three unknowns, the roll, pitch, and yaw. Therefore, if the A, B, and C values can be estimated separately from the observed frequency shifts (by Fourier analysis of the ΔF over one or more instrument scans), it is possible to obtain a unique solution for the attitude. In practice, since the yaw dependence is very weak, it is useful to constrain the yaw value and solve explicitly only for the roll and pitch. The option to compute either the yaw-constrained 2-component attitude solution or the full "free" solution for all three attitude components is maintained in the SeaWinds echo tracking algorithm implementation.

INPUTS

| | |
|--------------------|--|
| s_factor | An array of coefficients, each of which, when applied to the earth's surface elevation at a target location, determine the net fluctuation in the frequency shift of the scatterometer signal. |
| bb_freq_off_corr | A correction to the calculated change in baseband frequency which is used to determine the X Factor value for each echo measurement. |
| pulse_dim | The number of scatterometer pulses recorded in a telemetry frame. |
| echo_track_start | The sequential number of the telemetry frame record in each data partition where the Level 1B Processor begins to run the echo track algorithm to generate attitude data |
| echo_track_stop | The sequential number of the telemetry frame record in each data partition where the Level 1B Processor completes running the echo track algorithm to generate attitude data. |
| track_echo | Flag that indicates whether the echo tracking option is active. |
| rev_orbit_period | The approximate time between two consecutive ascending node crossings in the spacecraft orbital path. |
| true_cal_pulse_pos | The index of the loopback calibration in the calibration pulse sequence. |
| frame_qual_flag | Bit flags which indicate the character and the quality of the data acquired within a particular telemetry frame. |
| pulse_qual_flag | Bit flags which specify whether the quality of data for a particular pulse within a telemetry frame is reliable. |
| frame_inst_status | Bit flags which indicate the status of the SeaWinds instrument over the time span of a single telemetry frame. |
| frame_time_secs | The time which the SeaWinds Command and Data Subsystem (CDS) assigns to the telemetry data packet. |

| | |
|-----------------------|--|
| orbit_time | The time of each telemetry data packet relative to the time of the most recent ascending node crossing. |
| power_dn | The combined signal and noise power measured by the SeaWinds instrument. |
| doppler_orbit_step | The Doppler orbit step that is active at the time of the 100th pulse in the telemetry frame. |
| prf_orbit_step_change | Indicator of a change of the Doppler orbit step within a telemetry frame. The value designates the first pulse within the telemetry frame which is in the specified doppler_orbit_step for the associated telemetry frame. |
| range_gate_a_width | The period of time when the SeaWinds Scatterometer Electronic Subsystem (SES) receiver range gate is open to measure the echo power of pulses transmitted via the inner antenna beam. |
| range_gate_b_width | The period of time when the SeaWinds Scatterometer Electronic Subsystem (SES) receiver range gate is open to measure the echo power of pulses transmitted via the outer antenna beam. |
| pulse_width | The commanded duration of the pulse transmission for both channels A and B. |
| prf_cycle_time | The commanded time period between sequential pulse transmissions as a data number. |
| antenna_position | The antenna position as indicated by the SeaWinds Antenna Subsystem (SAS) at the falling edge of the first measurement pulse in the telemetry frame in data number units. |
| orb_smaj_axis | The length of the semimajor axis of the ADEOS II or QuikSCAT spacecraft orbit. |
| gap_structure | A structure that contains the data which are relevant to processing gaps in the Level Processors. |

OUTPUTS

| | |
|----------------------------|---|
| echo_track_attitude_struct | This data structure contains two major elements, num_attitude_recs and the Attitude structure. Num_attitude_recs specifies the number of available attitude records. Attitude is the data structure which contains one full set of attitude data. |
|----------------------------|---|

CONSTANTS

| | |
|---------------------|--|
| global_constants | A table of established physical or mathematical constants. |
| l1b_constants | A table that contains constants specific to the L1B Processor. |
| telemetry_constants | This table contains the run time constants required by the SeaWinds/QuikSCAT Processors and Preprocessors that manipulate telemetry and housekeeping data. |
| cell_geom_constants | This data structure contains a set of constants that are required to calculate the location of sigma0 cells and slices on the earth's surface. |

PROCESSING

For each input frame:

- Step 1. Call `Manage_Echo_Track_Frames` (L1B.5.2.0) to initialize and manage the echo track frame buffering.
- Step 2. Interpolate the ephemeris at the frame time to get the spacecraft state vector.
- Step 3. Compute the spacecraft nadir location.
- Step 4. Get the nearest attitude from the input attitude file. The input attitude is only used to obtain a better approximation of the footprint locations for the pulses; in the case where the input attitude data is absent or untrustworthy, use `[roll,pitch,yaw] = [0,0,0]` for the input attitude.
- Step 5. Compute other once-per-frame parameters used in the cell geometry.
- Step 6. Call `Determine_Echo_Track_Parameters` (L1B.5.3.1) to compute the spacecraft orbit radius and the radius of the earth ellipsoid at the nadir.
- Step 7. Compute the components of the local (S,T,U) coordinate system.
- Step 8. Compute the attitude rotation matrix (includes both instrument mount errors and input attitude errors).
- Step 9. Compute the relative velocity of the spacecraft with respect to the nadir location.

For each pulse in a frame:

- Step 10. Call `Ascertain_Beam_Parameters` (L1B.5.3.1) to determine various beam-related parameters (beam index, antenna elevation angle, range gate width, slant range) for the pulse.
- Step 11. Compute the antenna azimuth.
- Step 12. Compute the doppler parameters for the pulse.
- Step 13. Compute the components of the boresight vector to the pulse.
- Step 14. Transform the spacecraft state vector into the local coordinate system.
- Step 15. Compute the boresight vector components in the geocentric coordinate system.
- Step 16. Locate the pulse footprint on the Earth's surface in the geocentric system.
- Step 17. Compute the vector dot product of the local velocity vector and the local boresight vector.
- Step 18. Call `Acquire_Echo_Track_Matrix` (L1B.5.3.0) to compute the coefficients of the $\Delta F [A,B,C]$ vs. `[roll,pitch,yaw]` system of equations and the frequency shift ΔF for each pulse.

For each input frame:

- Step 19. Call `Track_Frame_Times` (L1B.5.2.3) to determine if there is enough buffered data to compute a new attitude set.

When sufficient data have been buffered:

- Step 20. Call `Calculate_Echo_Track_Attitude` (L1B.5.4.0) to solve the $\Delta F [A,B,C]$ vs. `[roll,pitch,yaw]` system of equations for the estimated attitude.

AUXILIARY DATA

| | |
|----------------------|---|
| doppler_range_delay | The Doppler Range Delay table contains two pairs of identical data structures. One pair contains the tabular data used to calculate commanded Doppler frequencies, the second pair contains the tabular data used to calculate range delays. |
| elevation | The representative elevation of the earth's surface for a roughly quadrilateral area that extends for 1/4 of a degree on each side. |
| rev_attitude_struct | This data structure contains two major elements, num_attitude_recs and the Attitude structure. Num_attitude_recs specifies the number of available attitude records. Attitude is the data structure which contains one full set of attitude data. |
| rev_ephemeris_struct | This data structure contains four major elements, num_ephemeris_recs, the Ephemeris structure, gap_count, and the Ancillary_Gap structure. Num_ephemeris_recs specifies the number of available ephemeris records. Ephemeris is the data structure which contains one full set of ephemeris data. |

REFERENCES

- [1] C. Wu, "SeaWinds Attitude Compensation and Determination", JPL IOM 3340-97-04CW, March 6, 1997.
- [2] R. S. Dunbar, "SeaWinds Attitude Determination from Echo Frequency Shifts", JPL IOM 3340-97-01RSD, June 10, 1997.

SeaWinds Algorithm Specification

TITLE: Manage Echo Track Frames
SUBMODULE: Manage Echo Track Frames
MODULE: Track Echo Signal
CODE: L1B.5.2.0
VERSION: 1.0
AUTHOR: R. Scott Dunbar, S. Vincent Hsiao, Philip S. Callahan
SUBROUTINE: Manage_Echo_Track_Frames
LANGUAGE: FORTRAN 90
HERITAGE: None

L1B.5.2.0 Manage_Echo_Track_Frames

PURPOSE

This subroutine tests input telemetry frames as they are used in echo tracking. Each echo track attitude is based on a series of more or less contiguous telemetry frames, for which the data required for echo tracking is stored in a buffer. The nominal number of frames in an echo track buffer is a runtime constant. The number of telemetry frames actually used (and usable) in an echo track block may change, however, if data gaps are located in the input data set, or if the operator specifies data skips.

INPUTS

| | |
|-----------------|---|
| echo_track_stop | The sequential number of the telemetry frame record in each data partition where the Level 1B Processor completes running the echo track algorithm to generate attitude data. |
| frame_time_secs | The time which the SeaWinds Command and Data Subsystem (CDS) assigns to the telemetry data packet. |

INPUTS/OUTPUTS

| | |
|---------------------------------|--|
| first_frame_in_partition_frames | A logical variable used to determine whether the current frame is the first one in the data set. |
| first_frame_in_partition_gaps | A logical variable used to determine whether the current frame is the first one in the data set. |
| track_echo_gap_structure | A structure that contains the data which are relevant to processing gaps in |

the echo tracking module of the Level Processors.

total_echo_track_frames

The actual number of telemetry frames that the Level 1B Processor uses to calculate an echo tracked attitude.

echo_track_block_abbreviated

A logical variable that indicates a data gap has significant effects on the echo track calculation.

last_echo_track_frame

A logical variable that indicates whether the current telemetry frame is the final frame in the set used to calculate an attitude measure based on the echo tracking algorithm.

skip_frame

A logical variable which indicates whether or not the current telemetry frame should be processed.

frame_number

This index specifies the active rev telemetry frame in Level 1A processing.

max_num_contiguous_frames

The maximum number of frames found within a predefined block of frames for deriving an echo track attitude record.

num_contiguous_frames

The number of frames found within a predefined block of frames that are next to one another.

used_frame

Flag indicating that a processed frame was used for deriving echo track attitude data.

The following are passed to Initialize_Echo_Tracking (L1B.5.2.1) when needed:

uf, vf, wf

The three elements of the "measurement" column vector for the least-squares estimator for attitude.

uu,uv,uw,
vv,vw,ww

The six unique matrix elements of the least-squares estimator for attitude.

total_frame_time_secs

The sum of the frame_time_secs measurements for all of the telemetry frames that contribute to a single echo tracking measurement.

CONSTANTS

| | |
|---------------|--|
| lib_constants | A table that contains constants specific to the LIB Processor. The key constant here is the nominal size of the echo track buffer (number of frames used in the buffer). |
|---------------|--|

PROCESSING

- Step 1. Under nominal conditions, increment the frame_number. However, if the previous call to Manage_Echo_Track_Frames indicated a sufficiently large gap to generate an abbreviated echo track attitude, the current call will not increment the frame_number nor check any of the other conditions related to that frame. Furthermore, if partitioning of the science data is employed (the nominal case), the frame number is not incremented at the beginning of a data partition.
- Step 2. If the current frame is the first in an echo track buffer, or if the previous telemetry frame is used to calculate an echo track attitude measurement, declare the current telemetry frame as the first frame in a new echo track block. The current frame time becomes the time against which to test subsequent frame times.
- Step 3. The summation variables that are used to calculate echo tracked attitudes are initialized by Initialize_Echo_Tracking (LIB.5.2.1). The contiguous frames counters are reset.
- Step 4. Determine whether the current frame is within a user specified gap. If it is, set the logical variable skip_frame to .TRUE. and return to Track_Echo_Signal (LIB.5.1.0).
- Step 5. Check the current frame time against the time of the first frame in the buffer. If the time difference is too large, set skip_frame to TRUE, echo_track_block_abbreviated to TRUE, and last_echo_track_frame to TRUE. Since skip_frame is TRUE, the subsequent echo track code will not include the current telemetry frame in the echo track calculation. Since echo_track_block_abbreviated is TRUE, the next call to Manage_Echo_Track_Frames will not increment the frame_number. Thus, the current telemetry frame will be included in the next echo track buffer.
- Step 6. If the current frame is the last one in the collection of frames being used to derive an attitude record, or if the current frame is the last one in the current partition, set the logical variable last_echo_track_frame to TRUE. This flag indicates to subsequent code that there is sufficient data in the buffer to calculate a new attitude record.

SeaWinds Algorithm Specification

TITLE: Initialize Echo Tracking
SUBMODULE: Manage Echo Track Frames
MODULE: Track Echo Signal
CODE: L1B.5.2.1
VERSION: 1.0
AUTHOR: R. Scott Dunbar, S. Vincent Hsiao, Philip S. Callahan
SUBROUTINE: Initialize_Echo_Track
LANGUAGE: FORTRAN 90
HERITAGE: None

L1B.5.2.1 Initialize_Echo_Track

PURPOSE

This subroutine initializes the summation variables that are used for each block of telemetry frames that generate an echo track attitude measurement.

INPUTS/OUTPUTS

| | |
|-------------------------|--|
| total_echo_track_frames | The actual number of telemetry frames that the Level 1B Processor uses to calculate an echo tracked attitude. |
| total_frame_time_secs | The sum of the frame_time_secs measurements for all of the telemetry frames that contribute to a single echo tracking measurement. |
| uf, vf, wf | The three elements of the "measurement" column vector for the least-squares estimator for attitude. |
| uu,uv,uw, vv,vw,ww | The six unique matrix elements of the least-squares estimator for attitude. |

PROCESSING

Step 1. Set the values of total_frame_time_secs and total_echo_track_frames to zero.

Step 2. Set the values of the measurement vector elements uf, vf, wf to zero.

Step 3. Set the values of the normal matrix elements uu,uv,uw,vv,vw,ww to zero.

SeaWinds Algorithm Specification

TITLE: Track Frame Times
SUBMODULE: Manage Echo Track Frames
MODULE: Track Echo Signal
CODE: L1B.5.2.2
VERSION: 1.0
AUTHOR: R. Scott Dunbar, S. Vincent Hsiao, Philip S. Callahan
SUBROUTINE: Track_Frame_Times
LANGUAGE: FORTRAN 90
HERITAGE: None

L1B.5.2.2 Track_Frame_Times

PURPOSE

This routine sums the frame times of the telemetry frames in the current echo tracking buffer and counts the number of frames in the buffer. These data are used to determine if there is sufficient data to produce the next attitude set from the echo tracking algorithm.

INPUTS

| | |
|-----------------|---|
| frame_time_secs | The time tag (in seconds from 1/1/1993) assigned to the input Level 1A frame. |
|-----------------|---|

INPUT/OUTPUTS

| | |
|-------------------------|---|
| total_echo_track_frames | The cumulative total of frames in the echo tracking buffer. |
| total_frame_time_secs | The sum of the frame times of the frames in the echo tracking buffer. |

PROCESSING

Step 1. Add the current frame_time_secs to the cumulative total_frame_time_secs.

Step 2. Increment the value of the frame counter (total_echo_track_frames).

SeaWinds Algorithm Specification

TITLE: Acquire Echo Track Matrix
SUBMODULE: Acquire Echo Track Matrix
MODULE: Track Echo Signal
CODE: L1B.5.3.0
VERSION: 1.0
AUTHOR: R. Scott Dunbar, S. Vincent Hsiao, Philip S. Callahan
SUBROUTINE: Acquire_Echo_Track_Matrix
LANGUAGE: FORTRAN 90
HERITAGE: None

L1B.5.3.0 Acquire_Echo_Track_Matrix

PURPOSE

This routine accumulates the elements of the attitude determination matrix used in the echo-tracking algorithm.

BACKGROUND

The application of the echo tracking algorithm for attitude determination depends on the changes in the observed frequency of the peak return of the SeaWinds echo signal. This attitude-induced frequency shift is composed of both a doppler component and a range component, the latter due to the chirped transmit pulse of the instrument in its high-resolution mode. The frequency shift can be formally expressed as:

$$\Delta F = \Delta F_R + \Delta F_D = (2\mu/c) \delta \mathbf{R} \cdot \mathbf{R}_u + (2F/c) d\mathbf{R}_u \cdot \mathbf{V}_r$$

where F is the transmit frequency in Hz, μ is the chirp rate in Hz/sec, \mathbf{R} and $\delta \mathbf{R}$ are the slant range vector and its attitude-induced shift, \mathbf{R}_u and $\delta \mathbf{R}_u$ are their respective unit vectors, and \mathbf{V}_r is the relative velocity of the spacecraft and the target location. The total ΔF is thus a function of the spacecraft location and velocity, the target location, and the attitude of the spacecraft. Given the spacecraft orbit, the frequency shift can be expressed as a function of the scan azimuth θ in the form:

$$\Delta F = A \cos \theta + B \sin \theta + C$$

where the coefficients A , B , and C are in turn functions of the orbit (spacecraft location and relative velocity), instrument look angle, and attitude.

The frequency shift ΔF as used here is the deviation of the actual frequency of the peak

signal return from that expected in the absence of attitude errors. To measure these deviations from the data, we use the power measurements for the slices (in data numbers, dn) in each pulse to determine the power centroid and its frequency within the echo bandwidth and compare it to the expected value. The centroid is a weighted average of the power dn values P_i over all slices i :

$$C_{\text{power}} = \sum C f_i P_i^4 / \sum P_i^4$$

where the $C f_i$ are a set of constant weights (including a slice number factor of 1/10). The measured frequency shift ΔF is then given by:

$$\Delta F = -C_{\text{power}} B_{\text{echo}} + \Delta F_o + \delta_{\text{fft}}/2$$

where B_{echo} is the echo bandwidth of the receiver over the center 10 slices, ΔF_o is the "expected" frequency shift assuming no attitude error, and δ_{fft} is the FFT bin width of the receiver.

The analysis by Dunbar [1] provided formulas for the A, B, and C coefficients in terms of the attitude errors ρ , ϕ , and ψ (roll, pitch, and yaw, respectively), the antenna look angle α , and the components of the relative velocity \mathbf{V}_{rel} of the spacecraft with respect to the target (in the local spacecraft coordinate system):

$$A = [(2F/c) V_{\text{rel},y} \sin \alpha] \psi - (2/c)[\mu\gamma + F V_{\text{rel},z} \sin \alpha] \phi$$

$$B = [-(2F/c) V_{\text{rel},x} \sin \alpha] \psi + (2/c)[\mu\gamma + F V_{\text{rel},z} \sin \alpha] \rho$$

$$C = [(2F/c) V_{\text{rel},x} \cos \alpha] \phi - [(2F/c) V_{\text{rel},y} \cos \alpha] \rho$$

where F is the transmit frequency, c is the speed of light in vacuum, μ is the chirp rate, and the geometric factor γ is given by:

$$\gamma = r \sin \alpha \{ 1 - \cos \alpha [(R_e^2/r^2) - \sin^2 \alpha]^{-1/2} \}$$

in which r is the spacecraft orbit radius at the observation time and R_e is the radius of the Earth. These formulas are based in part on the assumption that the attitude errors are small angles.

Rearrangement of the terms in the ΔF equation yields a linear equation in the attitude errors for the i^{th} pulse of the form:

$$\Delta F_i = U_i \rho + V_i \phi + W_i \psi$$

The formal least-squares solution to the linear system obtained by accumulating measurements over several scans is:

$$[\rho \ \phi \ \psi] = ([U_i \ V_i \ W_i]^T [U_i \ V_i \ W_i])^{-1} [U_i \ V_i \ W_i]^T \Delta F_i$$

The matrix elements that are required are thus:

Elements of the vector $[U_i \ V_i \ W_i]^T \Delta F_i$:

$$uf = \sum U_i \Delta F_i \qquad vf = \sum V_i \Delta F_i \qquad wf = \sum W_i \Delta F_i$$

Elements of the symmetric $[U_i \ V_i \ W_i]^T [U_i \ V_i \ W_i]$ matrix:

$$\begin{aligned} uu &= \sum U_i U_i & uv &= \sum U_i V_i & uw &= \sum U_i W_i \\ vv &= \sum V_i V_i & vw &= \sum V_i W_i & ww &= \sum W_i W_i \end{aligned}$$

Once these coefficients have been accumulated over sufficient data to produce an accurate solution, the estimates of the attitude errors can be computed.

INPUTS

| | |
|--------------------------|--|
| spacecraft_orbit_radius | The radius of the spacecraft orbit at the current spacecraft position. |
| earth_radius | The distance from the sea level to the center of the earth at the current spacecraft position. |
| slice_res_index | The array index which corresponds to the current effective gate width. |
| beam_index | A number which identifies one of the two SeaWinds antenna beams. |
| local_velocity | The vector that describes the velocity of the spacecraft in the spacecraft local coordinate system. The local coordinate system is centered at the orbital position of the spacecraft. |
| slant_range | The distance from the spacecraft to the center of the sigma0 cell on the earth's surface. |
| rangedelay_center_factor | A correction factor applied to the commanded range gate delay that centers the echo pulse within the range gate time span. |
| commanded_rangedelay | The range gate delay for a particular scatterometer pulse based on the contents of the Doppler Range Table. |
| commanded_doppler_freq | The Doppler frequency of a particular scatterometer pulse based on the contents of the Doppler Range Table. |
| calculated_doppler_freq | The Doppler frequency for a particular scatterometer pulse based on Level 1B Processor calculations. |
| power_dn | The combined signal and noise power measured by the SeaWinds instrument. |
| pulse_orbit_time | The time of the falling edge of a scatterometer pulse relative to a single orbital cycle. |
| rev_orbit_period | The approximate time between two consecutive ascending node crossings in the spacecraft orbital path. |
| cell_lat | The geodetic latitude of the center of a whole pulse sigma0 cell. |
| cell_lon | The longitude of the center of a whole pulse sigma0 cell. |

| | |
|------------------|--|
| elevation | The representative elevation of the earth's surface for a roughly quadrilateral area that extends for 1/4 of a degree on each side. |
| s_factor | An array of coefficients, each of which, when applied to the earth's surface elevation at a target location, determine the net fluctuation in the frequency shift of the scatterometer signal. |
| bb_freq_off_corr | A correction to the calculated change in baseband frequency which is used to determine the X Factor value for each echo measurement. |
| antenna_azimuth | The antenna azimuth as indicated by the SeaWinds Antenna Subsystem (SAS) at the falling edge of each measurement pulse . |

OUTPUTS

| | |
|-----------------------|---|
| uf, vf, wf | The three elements of the "measurement" column vector for the least-squares estimator for attitude. |
| uu,uv,uw, vv,vw,ww | The six unique matrix elements of the least-squares estimator for attitude. |

CONSTANTS

| | |
|---------------------|--|
| global_constants | The entries in the Global Constants table are well established physical, mathematical and astronomical constants. |
| 11b_constants | This table contains the run time constants required by the Level 1B Processor. |
| cell_geom_constants | This data structure contains a set of constants that are required to calculate the location of sigma0 cells and slices on the earth's surface. |

PROCESSING

Step 1. For the given parameters of the pulse computed previously in the geometric and orbital computations, compute the values of the U_i , V_i , and W_i coefficients of the ΔF equation.

Step 2. Form the products $U_i \Delta F_i$, $V_i \Delta F_i$, and $W_i \Delta F_i$ and add them to their respective measurement vector elements uf, vf, and wf.

Step 3. Compute the terms $U_i U_i$, $U_i V_i$, $U_i W_i$, $V_i V_i$, $V_i W_i$, $W_i W_i$, and add them to their respective matrix elements uu, uv, uw, vv, vw, and ww.

REFERENCES

- [1] R. S. Dunbar, "SeaWinds Attitude Determination from Echo Frequency Shifts", JPL IOM 3340-97-01RSD, June 10, 1997.

SeaWinds Algorithm Specification

TITLE: Ascertain Beam Parameters
SUBMODULE: Acquire Echo Track Matrix
MODULE: Track Echo Signal
CODE: L1B.5.3.1
VERSION: 1.0
AUTHOR: R. Scott Dunbar, S. Vincent Hsiao, Philip S. Callahan
SUBROUTINE: Ascertain_Beam_Parameters
LANGUAGE: FORTRAN 90
HERITAGE: None

L1B.5.3.1 Ascertain_Beam_Parameters

PURPOSE

This subroutine determines beam based parameters that are required to use scatterometer beams in the echo tracking algorithm. The subroutine operates on each pulse in a scatterometer frame to identify which parameters are applicable to the pulse.

INPUTS

| | |
|--------------------------|---|
| first_pulse_beam | A number that identifies whether the first pulse in a telemetry frame was generated by the inner or the outer scatterometer beam. |
| i_pulse | A counter which specifies a particular pulse within a telemetry frame (1-100). |
| range_gate_a_width | The period of time when the SeaWinds Scatterometer Electronic Subsystem (SES) receiver range gate is open to measure the echo power of pulses transmitted via the inner antenna beam. |
| range_gate_b_width | The period of time when the SeaWinds Scatterometer Electronic Subsystem (SES) receiver range gate is open to measure the echo power of pulses transmitted via the outer antenna beam. |
| slant_range_center_inner | The distance from the spacecraft to the center of the sigma0 cell generated by the SeaWinds instrument inner beam on the earth's surface. |
| slant_range_center_outer | The distance from the spacecraft to the center of the sigma0 cell generated by the SeaWinds instrument outer beam on the earth's surface. |

OUTPUTS

| | |
|-------------------|--|
| beam_index | A number which identifies one of the two SeaWinds antenna beams. |
| antenna_elevation | The angle of the maximum gain of the SeaWinds antenna with |

respect to the spacecraft Z axis.

range_gate_width_dn The period of time when the SeaWinds Scatterometer Electronic Subsystem (SES) receiver range gate is open to measure the echo power for the active scatterometer beam.

slant_range_center The distance from the spacecraft to the center of the sigma0 cell generated by the SeaWinds instrument on the earth's surface.

CONSTANTS

l1b_constants A table that contains constants specific to the L1B Processor.

PROCESSING

Step 1. Determine the relative pulse position within the frame. Since the SeaWinds pulses alternate between the inner and outer beams, we only need to determine if the pulse position is odd [$\text{mod}(i_pulse, 2) = 1$] or even [$\text{mod}(i_pulse, 2) = 0$].

Step 2. If the pulse position is odd and the first pulse of the frame was generated by the inner beam, or if the pulse position is even and the first pulse of the frame was generated by the outer beam, then:

```
beam_index = inner_beam_index
antenna_elevation = inner beam antenna look angle (from L1B constants)
range_gate_width = range_gate_a_width (for inner beam)
slant_range_center = slant range to center for an inner beam cell
```

Else (the pulse is from the outer beam):

```
beam_index = outer_beam_index
antenna_elevation = outer beam antenna look angle (from L1B constants)
range_gate_width = range_gate_b_width (for outer beam)
slant_range_center = slant range to center for an outer beam cell
```

SeaWinds Algorithm Specification

TITLE: Determine Echo Track Parameters
SUBMODULE: Acquire Echo Track Matrix
MODULE: Track Echo Signal
CODE: L1B.5.3.2
VERSION: 1.0
AUTHOR: R. Scott Dunbar, S. Vincent Hsiao, Philip S. Callahan
SUBROUTINE: Determine_Echo_Track_Params
LANGUAGE: FORTRAN 90
HERITAGE: None

L1B.5.3.2 Determine_Echo_Track_Params

PURPOSE

This routine calculates the spacecraft orbit radius and the earth radius at the spacecraft nadir, which are needed for later echo tracking computations. These values are computed once per frame.

INPUTS

spacecraft_lat The geodetic latitude of the spacecraft nadir.
spacecraft_pos_vel The spacecraft state vector (position and velocity components).

OUTPUTS

spacecraft_orbit_radius The geocentric radius of the spacecraft orbit at the current spacecraft position.
earth_radius The geocentric radius of the earth ellipsoid at the spacecraft nadir location.

CONSTANTS

global_constants The entries in the Global Constants table are well established physical, mathematical and astronomical constants. The key constants used here are:

earth_equatorial_radius = 6378136.3 meters
earth_flattening = 0.0033528131778969144

PROCESSING

Step 1. Compute the spacecraft orbit radius from the state vector position components (x,y,z):

$$\text{spacecraft_orbit_radius} = (x^2 + y^2 + z^2)^{1/2}$$

Step 2. Compute the radius of the earth ellipsoid at the spacecraft nadir location:

$$R(\text{spacecraft_lat}) = R_{\text{earth}} * (1 - \text{flat} * \sin(\text{spacecraft_lat}))$$

SeaWinds Algorithm Specification

TITLE: Calculate Echo Track Attitude
SUBMODULE: Calculate Echo Track Attitude
MODULE: Track Echo Signal
CODE: L1B.5.4.0
VERSION: 1.0
AUTHOR: R. Scott Dunbar, S. Vincent Hsiao, Philip S. Callahan
SUBROUTINE: Calculate_Echo_Track_Attitude
LANGUAGE: FORTRAN 90
HERITAGE: None

L1B.5.4.0 Calculate_Echo_Track_Attitude

PURPOSE

This routine computes the estimated attitude state from the accumulated echo-tracking data.

BACKGROUND

The application of the echo tracking algorithm for attitude determination depends on the changes in the observed frequency of the peak return of the SeaWinds echo signal. The total ΔF is thus a function of the spacecraft location and velocity, the target location, and the attitude of the spacecraft. Given the spacecraft orbit, the frequency shift can be expressed as a function of the scan azimuth θ in the form:

$$\Delta F = A \cos \theta + B \sin \theta + C$$

where the coefficients A, B, and C are in turn functions of the orbit (spacecraft location and relative velocity), instrument look angle, and attitude.

Given the expressions for the A, B, and C coefficients, the terms in the ΔF equation can be rearranged to yield a linear equation in the attitude errors for the i^{th} pulse of the form:

$$\Delta F_i = U_i \rho + V_i \phi + W_i \psi$$

where ρ , ϕ , and ψ are the roll, pitch, and yaw attitude error angles, respectively.

The formal least-squares solution to this linear system, obtained by accumulating measurements over several scans is:

$$[\rho \ \phi \ \psi] = ([U_i \ V_i \ W_i]^T [U_i \ V_i \ W_i])^{-1} [U_i \ V_i \ W_i]^T \Delta F_i$$

The elements of the normal matrix $[U_i \ V_i \ W_i]^T [U_i \ V_i \ W_i]$ and the measurement vector $[U_i \ V_i \ W_i]^T \Delta F_i$ are accumulated in the Acquire Echo Track Matrix subroutine (L1B.5.3.0). Once sufficient data has been accumulated, over at least several complete scans, the normal matrix can be inverted and multiplied by the measurement vector to yield the attitude error estimates.

Due to the weak dependence of the echo tracking data on yaw, we can constrain the solution to estimating only the roll and pitch by modifying the normal matrix and the measurement vector such that:

$$N_{ij} = [U_i \ V_i \ W_i]^T [U_i \ V_i \ W_i] \quad \begin{aligned} &= 0 \text{ for } i = 3 \text{ or } j = 3 \text{ when } i \neq j, \\ &= 1 \text{ for } i = j = 3 \end{aligned}$$

$$M_i = [U_i \ V_i \ W_i]^T \Delta F_i \quad \begin{aligned} &= \sum U_i \Delta F_i - \text{yaw_bias} * \sum U_i W_i \text{ for } i = 1 \\ &= \sum V_i \Delta F_i - \text{yaw_bias} * \sum V_i W_i \text{ for } i = 2 \\ &= 0 \text{ for } i = 3 \end{aligned}$$

The solution for roll and pitch proceeds with the inversion of the remaining 2x2 normal matrix and the 2x1 measurement vector.

This constrained solution is recommended for normal operational processing once the echo-tracked attitude determination is validated. It is expected that the yaw bias will be separately determined (from a larger quantity of data) during the calibration/validation phase post-launch.

INPUTS

| | |
|-------------------------|---|
| track_echo | Flag that indicates whether the echo tracking option is active, and in which mode – either the constrained 2-parameter [roll and pitch only] attitude solution or the free 3-parameter attitude solution. |
| total_echo_track_frames | The actual number of telemetry frames that the Level 1B Processor uses to calculate an echo tracked attitude. |
| total_frame_time_secs | The sum of the frame_time_secs measurements for all of the telemetry frames that contribute to a single echo tracking measurement. |
| uf, vf, wf | The three elements of the "measurement" column vector for the least-squares estimator for attitude. |
| uu,uv,uw, vv,vw,ww | The six unique matrix elements of the least-squares estimator for attitude. |

OUTPUTS

echo_track_attitude_struct This data structure contains two major elements, num_attitude_recs and the Attitude structure.

CONSTANTS

global_constants The entries in the Global Constants table are well established physical, mathematical and astronomical constants.

11b_constants This table contains the run time constants required by the Level 1B Processor.

PROCESSING

Step 1. Determine that there is sufficient contiguous data in the echo-tracking buffer to proceed to an attitude solution.

Step 2. Check which attitude solution mode is specified (do either step 3 or step 4).

Step 3. Constrained 2-parameter (roll, pitch) attitude solution:

- a) Allocate a 2x2 normal matrix and 2x1 measurement vector.
- b) Fill the 2x2 matrix as follows:

$$\begin{aligned} N_{11} &= uu &= \sum U_i U_i \\ N_{22} &= vv &= \sum V_i V_i \\ N_{12} = N_{21} &= uv &= \sum U_i V_i \end{aligned}$$

- c) Fill the 2x1 measurement vector M_i as follows:

$$\begin{aligned} M_1 &= uf - \text{yaw_bias} * uw = \sum U_i \Delta F_i - \text{yaw_bias} * \sum U_i W_i \\ M_2 &= vf - \text{yaw_bias} * vw = \sum V_i \Delta F_i - \text{yaw_bias} * \sum V_i W_i \end{aligned}$$

- d) Call Determine Singularity(L1B.5.4.1) to check that the normal matrix is invertible.

- e) Call Generate_Inverse_Matrix to invert the normal matrix.

- f) Multiply the inverted normal matrix by the measurement vector to generate the estimated roll and pitch; assign the yaw value to be equal to the value of the yaw_bias.

Step 4. Free 3-parameter (roll, pitch, yaw) attitude solution:

- a) Allocate a 3x3 normal matrix and 3x1 measurement vector.
- b) Fill the 3x3 matrix as follows:

$$\begin{aligned}
 N_{11} &= uu = \sum U_i U_i \\
 N_{22} &= vv = \sum V_i V_i \\
 N_{33} &= ww = \sum W_i W_i \\
 N_{12} = N_{21} &= uv = \sum U_i V_i \\
 N_{13} = N_{31} &= uw = \sum U_i W_i \\
 N_{23} = N_{32} &= vw = \sum V_i W_i
 \end{aligned}$$

- c) Fill the 3x1 measurement vector M_i as follows:

$$\begin{aligned}
 M_1 &= uf = \sum U_i \Delta F_i \\
 M_2 &= vf = \sum V_i \Delta F_i \\
 M_3 &= wf = \sum W_i \Delta F_i
 \end{aligned}$$

- d) Call Determine Singularity(L1B.5.4.1) to check that the normal matrix is invertible.
- e) Call Generate_Inverse_Matrix to invert the normal matrix.
- f) Multiply the inverted normal matrix by the measurement vector to generate the estimated roll, pitch, and yaw values.

SeaWinds Algorithm Specification

TITLE: Determine Singularity
SUBMODULE: Calculate Echo Track Attitude
MODULE: Track Echo Signal
CODE: L1B.5.4.1
VERSION: 1.0
AUTHOR: R. Scott Dunbar, S. Vincent Hsiao, Philip S. Callahan
SUBROUTINE: Determine_Singularity
LANGUAGE: FORTRAN 90
HERITAGE: None

L1B.5.4.1 Determine_Singularity

PURPOSE

This routine tests whether the attitude matrix accumulated in the echo tracking algorithm is singular, by computing and testing its determinant. If the determinant is zero (to within the machine accuracy given by $\epsilon = 10^{-9}$) then the matrix is singular and does not possess an inverse.

INPUTS

matrix An array of either 2 rows and columns (needed for the yaw-constrained case) or 3 rows and columns (needed for the unconstrained "free" solution).

OUTPUTS

is_singular A logical flag indicating whether the matrix is singular.

INTERNAL VARIABLES

work_matrix A 3x3 array used to compute the determinant of the input matrix. It is initialized to zero, and the elements of the input matrix are copied to the appropriate elements of the work_matrix. If the input matrix is a 2x2 (yaw-constrained case), the (3,3) element of the work_matrix is set to 1.

PROCESSING

Step 1. Determine the size of the input matrix (2x2 or 3x3).

Step 2. Initialize the work_matrix, setting all elements to zero.

Step 3. If the input matrix is 2x2, set work_matrix(3,3) = 1.

Step 4. Copy the elements of the input matrix into the work_matrix.

Step 5. Compute the determinant of the work matrix:

```
determinant = work_array(1,1)*[work_matrix(2,2)*work_matrix(3,3) -  
                               work_matrix(2,3)*work_matrix(3,2)]  
- work_array(2,1)*[work_matrix(1,2)*work_matrix(3,3) -  
                   work_matrix(1,3)*work_matrix(3,2)]  
+ work_array(3,1)*[work_matrix(1,2)*work_matrix(2,3) -  
                   work_matrix(1,3)*work_matrix(2,2)]
```

Step 6. Test the value of the determinant:

```
if abs(determinant) > e      :      is_singular = .false.  
else                        :      is_singular = .true.
```

The value of the flag is_singular is returned to the caller to indicate whether the matrix can be inverted or not.

COMMENTS

All computations should be done in double precision to maintain accuracy.

SeaWinds Scatterometer Calibration Smoothing Algorithm

Module L1B.6.0

ALGORITHM SPECIFICATIONS

AUTHOR: Barry H. Weiss
VERSION: 1.0
DATE: June 29, 2001

SeaWinds Scatterometer Calibration Smoothing Algorithm

MODULE (L1B.6.0)

I. MODULE OVERVIEW

Under nominal operating conditions, the SeaWinds instrument acquires a calibration sequence approximately once every 1.5 seconds. Each sequence consists of two calibration measurements. The first measurement is a loop back calibration, where the instrument feeds the energy from a standard pulse through attenuators to the receiver. The second is a load calibration, where the receiver records the ambient conditions when no signal is transmitted.

The telemetry denotes where calibration pulse sequences are located in the data stream. The Level 1A Processor locates each calibration pulse sequence. The Level 1A Processor then checks whether each earmarked calibration displays a reasonable spectral profile for a calibration pulse in the current resolution mode. If the calibration sequence appears to be correct, the Level 1A Processor copies the calibration pulse sequence, along with a few additional data elements that are required to interpret the calibration data, into a record in the Calibration Pulse Product.

A series of sensitivity studies by Lou and Liu [1] indicates that the use of time averaged calibration data can significantly reduce the error in σ_o estimates. Their study shows that the error in s_o reduces to an acceptable level of 0.2 dB for low signal to noise ratio conditions when a time average of 800 or more calibration values are used to estimate the bandwidth ratio of noise filter to echo filter. The level of error in s_o improves significantly when the signal to noise ratio is higher. Based on the work on Lou and Liu, the Level 1B Processor performs a time average on all calibrations before they are applied to the energy measurements in the echo filter and the noise filter.

Use of the Calibration Pulse Product simplifies the calibration smoothing process in the Level 1B Processor. Acceptable time averages of calibrations require access to an adequate range of data that both precede and follow each calibration record. Access to adequate input for the time averages becomes problematic when records are relatively close to data granule boundaries. To address the data boundary problem, a nominal run of the Level 1B Processor reads three Calibration Pulse Product files. One file represents the data granule that precedes the one being processed, the second file represents the data granule that is being processed, and the final file represents the data granule that follows the one being processed. The Level 1B Processor combines the data from these three files into a single buffer. This approach insures that the calibration data are smoothed equally well over the entire input data set and that any abrupt changes in calibration measure are detected.

II. FUNCTIONAL FLOW DESCRIPTION

The design divides the algorithm into three major segments.

Initialize Calibration Pulse Buffers for Smoothing

This segment populates the data buffers that are used to calculate the time averages of the loop back calibration and ambient load calibration. The Processor uses data in the Calibration Pulse Product file to populate the buffers. The Level 1B Processor calls this subroutine at the very beginning of the algorithmic process.

Apply Time Average to Calibration Pulses

This segment performs a rolling time average of the calibration pulse measurements. In the process, the code checks for abrupt changes in calibration measurements that might signal erroneous or undesirable instrument behavior. The code also insures that calibrations that are acquired in differing instrument modes are not combined in the data smoothing technique. The Level 1B Processor calls this subroutine at the very beginning of the algorithmic process.

Assign Appropriate Calibrations to Measurement Pulses

This module compares the time of transmission of each pulse against the representative times associated with each time averaged calibration record. The module assigns the calibration that best fits a time based criterion to each pulse. Subsequent code applies the assigned calibrations to the echo and noise energy measurements for the pulse to calculate the corresponding normalized radar cross section (σ_o). The Level 1B Processor calls this function at the beginning of module that calculates σ_o values.

SeaWinds Algorithm Specification

| | |
|--------------------|-------------------------------------|
| TITLE: | Initialize Calibration Time Average |
| SUBMODULE: | |
| MODULE: | Initialize Level 1B Processing |
| CODE: | L1B.6.1.1 |
| VERSION: | 1.0 |
| DATE: | 07/18/01 |
| AUTHOR: | Barry Weiss |
| SUBROUTINE: | Initialize_Cal_Pulse |
| LANGUAGE: | Fortran 90 |
| HERITAGE: | None |

L1B.6.1.1 Initialize_Cal_Pulse

PURPOSE

A study by Lou and Liu [1] indicates that the use of time averaged calibrations diminishes the error in σ_0 estimation. This algorithm prepares the set of calibration measurements for the implementation of a rolling time average. The number of data elements applied to each average meets the criteria recommended by Lou and Liu.

BACKGROUND

Initialize_Cal_Pulse populates the local variables that store the data required to calculate representative calibrations for the first pulse in the first telemetry frame of the input Level 1A data set. These variables include (1) the index of the calibration record that best approximates the first pulse in the first telemetry frame of the Level 1A product, (2) the indices that mark the beginning and the end of the time span over which the time average process will take place, (3) variables that store the sum of all of the good quality calibrations that are located within the time span and (4) variables that store the number of good quality calibrations that contribute to the sums in each time span. Subroutine Processing_Calibration_Data subsequently uses these variables to track and store the input data for calibration smoothing.

The logic in Initialize_Cal_Pulse presumes that the calibration pulse data are listed in temporal order. The Level 1A Processor logic generates the Calibration Pulse Product in temporal order, and thus guarantees this condition.

Initialize_Cal_Pulse employs a single technique to populate the variables that are used to smooth both calibration types. Initialize_Cal_Pulse assigns the index I_{mid} to the first record among the calibration data with a time that follows the transmission time of the first pulse in the first telemetry frame of the input Level 1A data set. The logic then employs three criteria to seek the full set of calibration data records that will be applied to the first time average calculation.

First, the calibrations that contribute to each time average must center about the calibration identified by index I_{mid} . Second, the basis for the centering criterion is the time associated with each calibration. Third, the process continuously identifies new data elements that will be used to smooth the calibration until the required number of elements has been incorporated.

The number of elements required to smooth calibrations depends on the type of calibration data being averaged. Data element Dim_{amb_load} in the Level 1B Constants specifies the nominal number of input entries required to smooth the load calibrations. In the current SeaWinds implementation, Dim_{amb_load} is 800. Data element Dim_{loop_back} in the Level 1B Constants specifies the nominal number of input entries required to smooth the loop back calibrations. In the current SeaWinds implementation, Dim_{loop_back} is 20. Both of these values are based on the recommendations of Lou and Liu [1].

To begin the search, `Initialize_Cal_Pulse` increments the boundary indices outward from the established center location.

$$I_{begin} = I_{mid} - 1 \quad (1)$$

$$I_{end} = I_{mid} \quad (2)$$

where

I_{begin} is the index of the first calibration record that will be used in the time average
 I_{end} is the index of the last calibration record that will be used in the time average

`Initialize_Cal_Pulse` then calculates the time difference between the calibration measures with index I_{mid} and the two calibration measures with indices I_{begin} and I_{end} .

$$Span_{mid_to_begin} = \text{abs}(T_{mid} - T_{begin}) \quad (3)$$

$$Span_{mid_to_end} = \text{abs}(T_{mid} - T_{end}) \quad (4)$$

where

T_{mid} time of the first calibration pulse entry that follows the time of the transmission of the first pulse in the first telemetry frame of the Level 1A Product
 T_{begin} time of the calibration pulse with index I_{begin}
 T_{end} time of the calibration pulse with index I_{end}
 $Span_{mid_to_begin}$ time span between the first calibration record that will be used in the time average and the assigned midpoint calibration
 $Span_{mid_to_end}$ time span between the last calibration record that will be used in the time average and the assigned midpoint calibration

`Initialize_Cal_Pulse` determines whether $Span_{mid_to_begin}$ or $Span_{mid_to_end}$ is larger. `Initialize_Cal_Pulse` then expands the endpoint index for the boundary that records the smaller time difference relative to the midpoint calibration.

$$\begin{aligned}
&\text{If Span}_{\text{mid_to_begin}} \text{ is less than Span}_{\text{mid_to_end}} \text{ then} \\
&\quad I_{\text{begin}} = I_{\text{begin}} - 1 \\
&\quad \text{Span}_{\text{mid_to_begin}} = \text{abs}(T_{\text{pulse}} - T_{\text{begin}}) \\
&\text{Else if } T_{\text{mid_to_begin}} \text{ is greater than or equal to } T_{\text{mid_to_end}} \text{ then} \\
&\quad I_{\text{end}} = I_{\text{end}} + 1 \\
&\quad \text{Span}_{\text{mid_to_end}} = \text{abs}(T_{\text{pulse}} - T_{\text{end}}) \\
&\text{End if}
\end{aligned} \tag{5}$$

Each iteration of this process expands the count of the number of elements in the time average. Thus, if the process is operating on load calibrations, then

$$\text{Num}_{\text{amb_load}} = \text{Num}_{\text{amb_load}} + 1 \tag{6}$$

where

$\text{Num}_{\text{amb_load}}$ is the number of load calibrations in the time average

This process continues until the number of load calibrations, or $\text{Num}_{\text{amb_load}}$, is equal to $\text{Dim}_{\text{amb_load}}$.

On the other hand, if the process is operating on loop back calibrations, then

$$\text{Num}_{\text{loop_back},i_{\text{pol}}} = \text{Num}_{\text{loop_back},i_{\text{pol}}} + 1 \tag{7}$$

where

$\text{Num}_{\text{loop_back},i_{\text{pol}}}$ is the number of loop back calibrations in the time average for the specified polarization

This process continues until the number of loop back calibrations for each beam polarization, or $\text{Num}_{\text{loop_back},i_{\text{pol}}}$, is equal to $\text{Dim}_{\text{loop_back}}$.

Under nominal conditions, this logic insures that the time average covers approximately the same time span before the midpoint calibration and after the midpoint calibration. The logic does not, however, require that the number of elements that precede the midpoint calibration is equal to the number of elements that follow the midpoint calibration.

Initialize_Cal_Pulse does adjust the inclusion of new elements into the time average if the logic detects that the boundaries have reached the endpoints of the data provided in the Calibration Pulse Product. Thus, if index I_{begin} reaches 1 before the number of elements reaches the specified target, Initialize_Cal_Pulse will continue to increment I_{end} until the target number of elements are included in the time average, regardless of the time difference. Likewise, if I_{end} reaches the final record in the Calibration Pulse Product before the number of elements reaches the specified target, Initialize_Cal_Pulse will continue to decrement I_{begin} until the target number of elements

are included in the time average, regardless of the time difference.

Once the boundaries of the set of calibrations that will contribute to the first time average have been located, the processing for ambient load and loop back calibrations varies.

The algorithm generates a total echo energy for each load calibration pulse within the time average boundaries by calculating the sum of the echo energy of the twelve slices in each load calibration record. The algorithm then sums the total echo energy for every load calibration pulse within the time average boundaries, and places that sum in data element $\text{Sum}_{E_amb_load}$. The algorithm also sums the total noise measure for all of the load calibrations within the time average boundaries. The subroutine passes this value to data element $\text{Sum}_{N_amb_load}$.

For loop back calibrations, the algorithm operates on the calibrations for each polarization separately. The algorithm first sums the echo energy of the twelve slices for each of the loop back calibrations within the time average boundaries. The process then checks whether the distribution of loop back calibrations for each polarization is relatively smooth. The algorithm sorts the loop back calibrations for each polarization relative to the total echo energy. Using the ordered calibrations, the algorithm calculates the mean of those calibrations that lie in the second and the third quartiles of the distribution. The algorithm assigns this mean to a test calibration value, or Test_{i_pol} .

$\text{Initialize_Cal_Pulse}$ loops through the loop back calibrations from I_{begin} to I_{end} once more. This time, the algorithm compares the relative change in value of each loop back calibration against the test value of Test_{i_pol} .

$$\text{Net_Change} = (\text{Echo}_{\text{loop_back},i_pol} - \text{Test}_{i_pol}) / \text{Test}_{i_pol} \quad (8)$$

where

$\text{Echo}_{\text{loop_back},i_pol}$ represents the total echo energy for each of the elements among the set of loop back calibrations that are candidates for the first time average calculation

Thus, if Net_Change is less than or equal to $\Delta_{\text{loop_back}}$, the algorithm adds $\text{Echo}_{\text{loop_back},i_pol}$ into the summation variable $\text{Sum}_{E_loop_back,i_pol}$. $\text{Sum}_{E_loop_back,i_pol}$ is the total echo energy over all twelve slices for all of the acceptable loop back calibration pulses of the specified polarization from index I_{begin} to index I_{end} . Likewise, the logic adds the corresponding loop back noise energy, or $\text{Noise}_{\text{loop_back},i_pol}$ into the summation variable $\text{Sum}_{N_loop_back,i_pol}$. $\text{Sum}_{N_loop_back,i_pol}$ is the total noise energy for all of the acceptable loop back calibration pulses of the specified polarization from index I_{begin} to index I_{end} .

On the other hand, if Net_Change is greater than $\Delta_{\text{loop_back}}$, $\text{Initialize_Cal_Pulse}$ does not include $\text{Echo}_{\text{loop_back},i_pol}$ in $\text{Sum}_{E_loop_back,i_pol}$ and does not include $\text{Noise}_{\text{loop_back},i_pol}$ in $\text{Sum}_{N_loop_back,i_pol}$. $\text{Initialize_Cal_Pulse}$ also decrements the counter $\text{Num}_{\text{loop_back},i_pol}$.

The current design presumes that the number load calibrations that are available for processing always exceeds $\text{Dim}_{\text{amb_load}}$. If the input data set is very small, fewer than $\text{Dim}_{\text{amb_load}}$ calibrations may be available. Before the processor begins to load the buffer, the logic checks whether the number of records in the calibration data buffer are sufficient. If the total number of calibration records is fewer than $\text{MIN_LOAD_CAL_BUFFER}$, the processor halts. The nominal value of $\text{MIN_LOAD_CAL_BUFFER}$ is 200. If the total number of calibration records is greater than $\text{MIN_LOAD_CAL_BUFFER}$ and is less than $\text{Dim}_{\text{amb_load}}$, the processor resets the value of $\text{Dim}_{\text{amb_load}}$ to be equal to $\text{MIN_LOAD_CAL_BUFFER}$.

INPUTS

The algorithm reads these input parameters from the Calibration Pulse Product:

| | |
|---|---|
| $\text{Noise}_{\text{amb_load}}$ | Noise energy of an ambient load scatterometer calibration |
| $\text{Echo}_{\text{amb_load}}$ | Echo energy of an ambient load scatterometer calibration |
| $\text{Noise}_{\text{loop_back},i,\text{pol}}$ | Noise energy of a loop back scatterometer calibration for the specified beam polarization |
| $\text{Echo}_{\text{loop_back},i,\text{pol}}$ | Echo energy of a loop back scatterometer calibration for the specified beam polarization |

The algorithm acquires this input parameter from the Level 1A Product:

T_{pulse} The time of the transmission of the first pulse in the input data set

OUTPUTS

| | |
|---|--|
| $\text{Num}_{\text{amb_load}}$ | The number of load calibrations that were identified for the first time average calculation |
| $\text{Num}_{\text{loop_back},i,\text{pol}}$ | The number of loop back calibrations that were identified for the first time average calculation for each polarization |
| I_{mid} | Index of the midpoint calibration that will be used to calculate the first time average |
| $I_{\text{amb_load_begin}}$ | Index of the first load calibration that will be used to calculate the first time average |
| $I_{\text{amb_load_end}}$ | Index of the final load calibration that will be used to calculate the first time average |
| $I_{\text{loop_back_begin},i,\text{pol}}$ | Index of the first loop back calibration of the specified polarization that will be used to calculate the first time average |
| $I_{\text{loop_back_end},i,\text{pol}}$ | Index of the final loop back calibration of the specified polarization that will be used to calculate the first time average |
| $\text{Test}_{i,\text{pol}}$ | Test value used to eliminate possible outlier values from the distribution of loop back calibrations |
| $\text{Sum}_{\text{E_amb_load}}$ | The sum of the echo energy over all of the slices for all of the load calibration pulses from index $I_{\text{amb_load_begin}}$ to index $I_{\text{amb_load_end}}$. |
| $\text{Sum}_{\text{N_mb_load}}$ | The sum of the noise energy for all of the load calibration pulses from index $I_{\text{amb_load_begin}}$ to index $I_{\text{amb_load_end}}$. |

| | |
|--|---|
| $\text{Sum}_{\text{E_loop_back},i_{\text{pol}}}$ | The sum of the echo energy over all of the slices for all of the loop back calibration pulses of the specified polarization from index $I_{\text{loop_back_begin},i_{\text{pol}}}$ to index $I_{\text{loop_back_end},i_{\text{pol}}}$. |
| $\text{Sum}_{\text{N_loop_back},i_{\text{pol}}}$ | The sum of the noise energy for all of the loop back calibration pulses of the specified polarization from index $I_{\text{loop_back_begin},i_{\text{pol}}}$ to index $I_{\text{loop_back_end},i_{\text{pol}}}$. |

CONSTANTS

All of the following constants are stored in the Level 1B Constants Table:

| | | |
|------------------------------------|---|--------|
| $\text{Dim}_{\text{amb_load}}$ | The prescribed number of elements to calculate a time average of load calibrations. | 800 |
| $\text{Dim}_{\text{loop_back}}$ | The prescribed number of elements to calculate a time average of loop back calibrations. | 20 |
| $\text{Delta}_{\text{loop_back}}$ | The maximum change that should appear between a given loop back calibration and the test calibration value. | 1.2 dB |

PROCESSING

- Step 1: Checks whether the number of available calibrations is greater than $\text{MIN_LOAD_CAL_BUFFER}$. If fewer than $\text{MIN_LOAD_CAL_BUFFER}$ calibrations are available, the processor exits.
- Step 2: Checks whether the number of available calibrations is less than $\text{Dim}_{\text{amb_load}}$. If fewer the $\text{Dim}_{\text{amb_load}}$ calibrations are available, reset the value of $\text{Dim}_{\text{amb_load}}$ to be equal to $\text{MIN_LOAD_CAL_BUFFER}$.
- Step 3: Locates the first record among the input calibration pulse data that contains a time that is later than the transmission time of the first pulse in the first telemetry frame of the input Level 1A data set. Assigns the index I_{mid} to that calibration pulse record.
- Step 4: Expands the range of indices of the load calibrations using equations (1) and (2). Then calculates the time span between I_{mid} and $I_{\text{amb_load_begin}}$ using equation (3). Calculates the time span between I_{mid} and $I_{\text{amb_load_end}}$ using equation (4).
- Step 5: Determines which time span is smaller. Increments the index of the boundary that records the smaller time span relative to the midpoint calibration using the logic in (5). Increments the number of load calibrations in the buffer using (6). Continues this process until the number of load calibrations that extend from $I_{\text{amb_load_begin}}$ to $I_{\text{amb_load_end}}$ is $\text{Dim}_{\text{amb_load}}$. This count includes the records with indices $I_{\text{amb_load_begin}}$ and $I_{\text{amb_load_end}}$.
- Step 6: Sums the echo energy for all of the slices in all of the load calibration pulses from $I_{\text{amb_load_begin}}$ to $I_{\text{amb_load_end}}$, and places the sum in $\text{Sum}_{\text{E_amb_load}}$.

- Step 7: Sums the noise energy for all of the load calibration pulses from $I_{amb_load_begin}$ to $I_{amb_load_end}$, and places the sum in $Sum_{N_amb_load}$.
- Step 8: Loops over both scatterometer beams. The first loop iteration processes the horizontal polarization scatterometer beam. The second loop iteration processes the vertical polarization scatterometer beam.
- Step 9: Expands the indices of the loop back calibration buffer for the current beam using equations (1) and (2). Then calculates the time span between I_{mid} and $I_{loop_back_begin,i_pol}$ using equation (3) and the time span between I_{mid} and $I_{loop_back_end,i_pol}$ using equation (4).
- Step 10: Determines which time span is smaller. Increments the index of the boundary that records the smaller time span relative to the midpoint calibration using the logic in (5). Increments the number of loop back calibrations for the appropriate beam polarization using (7). Continues until the number of loop back calibrations from index $I_{loop_back_begin,i_pol}$ to $I_{loop_back_end,i_pol}$ is equal to Dim_{loop_back} . This count includes the records with indices $I_{loop_back_begin,i_pol}$ and $I_{loop_back_end,i_pol}$.
- Step 11: Sums the echo energy for all of the slices in each of the loop back calibration pulses from $I_{loop_back_begin,i_pol}$ to $I_{loop_back_end,i_pol}$, and places each sum into an individual entry in the array $Echo_{loop_back,i_pol}$.
- Step 12: Sorts all of the loop back calibrations in the array $Echo_{loop_back,i_pol}$ between index $I_{loop_back_begin,i_pol}$ and $I_{loop_back_end,i_pol}$ in numerical order.
- Step 13: Calculates the mean of the elements in the second and third quartiles of the distribution of the loop back calibrations. Assigns that mean to $Test_{i_pol}$.
- Step 14: Tests each loop back calibration of the appropriate beam polarization from index $I_{loop_back_begin,i_pol}$ to $I_{loop_back_end,i_pol}$ against $Test_{i_pol}$ using equation (8). If Net_Change is less than or equal to Δ_{loop_back} , proceeds to step 14. If Net_Change is greater than Δ_{loop_back} , tests the next loop back calibration in the sequence.
- Step 15: Includes the sum of the echo energy for all of the slices of the loop back calibration pulse into summation variable $Sum_{E_loop_back,i_pol}$ for the appropriate polarization.
- Step 16: Includes the noise energy of the loop back calibration pulse into the summation variable $Sum_{N_loop_back,i_pol}$ for the appropriate polarization.

REFERENCE

- [1] Lou, S. and Liu, Y., SeaWinds/QuikSCAT High and Low Resolution On-orbit Calibration and Noise Subtraction, Interoffice Memorandum 3347-98-019, Jet Propulsion Laboratory, March 27, 1998.

SeaWinds Algorithm Specification

TITLE: Apply Time Average to Calibration Data
SUBMODULE:
MODULE: Initialize Level 1B Processing
CODE: L1B.6.1.2
VERSION: 1.0
DATE: 07/18/01
AUTHOR: Barry Weiss
SUBROUTINE: Process_Calibration_Data
LANGUAGE: Fortran 90
HERITAGE: None

L1B.6.1.2 Process_Calibration_Data

PURPOSE

Lou and Liu [1] demonstrated that time averaging calibrations before their use significantly reduces the noise in σ_0 measure. This algorithm implements a rolling time average of the instrument calibration measurements in order to meet those recommendations.

BACKGROUND

Initialize_Cal_Pulse populates the variables that are used to perform the time average of the instrument calibrations. That function introduces a set of array indices that define the range of the calibrations that are applied to each time average calculation. These indices include:

| | |
|---|---|
| $I_{\text{amb_load_begin}}$ | Index of the first load calibration used to calculate a rolling time average |
| $I_{\text{amb_load_end}}$ | Index of the final load calibration used to calculate a rolling time average |
| $I_{\text{loop_back_begin},i_{\text{pol}}}$ | Index of the first loop back calibration of the specified polarization used to calculate a rolling time average |
| $I_{\text{loop_back_end},i_{\text{pol}}}$ | Index of the final loop back calibration of the specified polarization used to calculate a rolling time average |

When Initialize_Cal_Pulse finishes processing, all of the data that are required to calculate time averaged calibrations for the very first elements in the Level 1A Product have been identified. The first functional steps in Process_Calibration_Data complete that process. The following equations calculate the calibration parameters that the Level 1B Processor subsequently uses to generate σ_0 :

$$\text{Mean}_{E_{\text{amb_load}}} = \text{Sum}_{E_{\text{amb_load}}} / \text{Num}_{\text{amb_load}} \quad (1)$$

$$\text{Mean}_{N_{\text{amb_load}}} = \text{Sum}_{N_{\text{amb_load}}} / \text{Num}_{\text{amb_load}} \quad (2)$$

$$\text{Mean}_{\text{E_loop_back},i,\text{pol}} = \text{Sum}_{\text{E_loop_back},i,\text{pol}} / \text{Num}_{\text{loop_back},i,\text{pol}} \quad (3)$$

$$\text{Mean}_{\text{N_loop_back},i,\text{pol}} = \text{Sum}_{\text{N_loop_back},i,\text{pol}} / \text{Num}_{\text{loop_back},i,\text{pol}} \quad (4)$$

$$T_{\text{time_avg}} = T_{\text{mid}} \quad (5)$$

$$\alpha = (\text{Mean}_{\text{N_amb_load}} / \text{Mean}_{\text{E_amb_load}}) / \text{Ratio}_{\text{rcv_gain_eu}} \quad (6)$$

$$\text{Cal}_{i,\text{pol}} = (a * \text{Mean}_{\text{E_loop_back},i,\text{pol}} - \text{Mean}_{\text{N_loop_back},i,\text{pol}} / \text{Ratio}_{\text{rcv_gain_eu}}) / (\alpha - 1) \quad (7)$$

where

$\text{Mean}_{\text{E_amb_load}}$ the mean of the echo energy of the load calibrations with indices that range from $I_{\text{amb_load_begin}}$ to $I_{\text{amb_load_end}}$.

$\text{Mean}_{\text{N_amb_load}}$ the mean of the noise energy of the load calibrations with indices that range from $I_{\text{amb_load_begin}}$ to $I_{\text{amb_load_end}}$.

$\text{Mean}_{\text{E_loop_back},i,\text{pol}}$ the mean of the echo energy of the loop back calibrations for the specified beam polarization with indices that range from $I_{\text{loop_back_begin},i,\text{pol}}$ to $I_{\text{loop_back_end},i,\text{pol}}$.

$\text{Mean}_{\text{N_loop_back},i,\text{pol}}$ the mean of the noise energy of the loop back calibrations for the specified beam polarization with indices that range from $I_{\text{loop_back_begin},i,\text{pol}}$ to $I_{\text{loop_back_end},i,\text{pol}}$.

$T_{\text{time_avg}}$ the representative time for a time averaged calibration entry.

α the bandwidth ratio

$\text{Ratio}_{\text{rcv_gain_eu}}$ the SeaWinds instrument receiver gain ratio in engineering units.

$\text{Cal}_{i,\text{pol}}$ the representative calibration value for the specified beam polarization.

If no problems were detected in the calculation of these parameters, the algorithm sets the value of the flag $\text{Cal}_{\text{usable},i,\text{pol}}$ for the corresponding calibration record to TRUE. If problems were detected, the algorithm sets $\text{Cal}_{\text{usable},i,\text{pol}}$ to FALSE. This flag indicates to subsequent code whether the corresponding calibration data listings ought to be used.

Finally, `Process_Calibration_Data` checks whether the loop back calibration with index $I_{\text{loop_back_begin},i,\text{pol}}$ and the loop back calibration with index I_{mid} were both acquired when the SeaWinds instrument was operating in Wind Observation Mode. If both elements were acquired when the instrument was in Wind Observation Mode, `Process_Calibration_Data` sets the flag $\text{Cal}_{\text{mode_consistent}}$ to TRUE. This flag is designed to locate calibrations that were acquired at the very beginning of a Wind Observation Mode sequence. The SeaWinds instrument subsystems power up gradually at the beginning of Wind Observation Mode. The resulting lack of steady state conditions is likely to generate unreliable time averaged calibrations. The flag $\text{Cal}_{\text{mode_consistent}}$ is used in subsequent code to denote any time averaged calibrations that have questionable value due to these conditions.

With the completion of the first time average calculation of calibration parameters, `Process_Calibration_Data` begins an iterative process. Within each iteration, the algorithm identifies the range of data that apply to the next calibration time average, and then calculates a set of representative calibration parameters that are based on the time averaging method.

The iterative process begins by modifying the first, last and midpoint data indices that were defined in the previous time average calculation. The algorithm increments the index of the midpoint calibration.

$$I_{mid} = I_{mid} + 1 \quad (8)$$

The new midpoint provides a basis for the calculation of an entirely new set of calibration parameters. The time of the calibration measure with index I_{mid} is T_{mid} . The algorithm assigns the time T_{mid} to variable T_{time_avg} . T_{time_avg} is the reference time that corresponds to the time averaged calibration variables that this algorithm generates.

Process_Calibration_Data then adjusts the range of the load calibration measures for the next time average calculation. Based on the new midpoint location, the routine recalculates the time difference between the load calibration measure with index I_{mid} and the two load calibration measures with indices $I_{amb_load_begin}$ and $I_{amb_load_end}$.

$$Span_{mid_to_begin} = abs(T_{mid} - T_{amb_load_begin}) \quad (9)$$

$$Span_{mid_to_end} = abs(T_{mid} - T_{amb_load_end}) \quad (10)$$

where

T_{mid} time of the calibration pulse entry that currently represents the midpoint of the time average buffer

$T_{amb_load_begin}$ time of the load calibration pulse with index I_{begin}

$T_{amb_load_end}$ time of the load calibration pulse with index I_{end}

$Span_{mid_to_begin}$ time span between the load calibration with index $I_{amb_load_begin}$ and the midpoint calibration

$Span_{mid_to_end}$ time span between the load calibration with index $I_{amb_load_end}$ and the midpoint calibration

Process_Calibration_Data then compares $Span_{mid_to_begin}$ to $Span_{mid_to_end}$. If $Span_{mid_to_end}$ is smaller than or equal to $Span_{mid_to_begin}$, the range of input calibration measures that apply to the time average must change. The algorithm removes the first data entry from the summation variables since that entry now falls outside of the data range:

$$Sum_{E_amb_load} = Sum_{E_amb_load} - Echo_{amb_load}(I_{amb_load_begin}) \quad (11)$$

$$Sum_{N_amb_load} = Sum_{N_amb_load} - Noise_{amb_load}(I_{amb_load_begin}) \quad (12)$$

where

$Echo_{amb_load}(I_{amb_load_begin})$ The echo energy over all of the slices of the load calibration with index $I_{amb_load_begin}$.

$Noise_{amb_load}(I_{amb_load_begin})$ The noise energy of the load calibration with index $I_{amb_load_begin}$.

The algorithm then updates the indices of the first and last elements that apply to the rolling time average.

$$I_{\text{amb_load_begin}} = I_{\text{amb_load_begin}} + 1 \quad (13)$$

$$I_{\text{amb_load_end}} = I_{\text{amb_load_end}} + 1 \quad (14)$$

Based on the new value of index $I_{\text{amb_load_end}}$, the algorithm includes the new calibration measures that now fall within range into the sums that are used to calculate the time averages:

$$\text{Sum}_{\text{E_amb_load}} = \text{Sum}_{\text{E_amb_load}} + \text{Echo}_{\text{amb_load}}(I_{\text{amb_load_end}}) \quad (15)$$

$$\text{Sum}_{\text{N_amb_load}} = \text{Sum}_{\text{N_amb_load}} + \text{Noise}_{\text{amb_load}}(I_{\text{amb_load_end}}) \quad (16)$$

where:

$\text{Echo}_{\text{amb_load}}(I_{\text{amb_load_end}})$ The echo energy over all of the slices of the load calibration with index $I_{\text{amb_load_end}}$.

$\text{Noise}_{\text{amb_load}}(I_{\text{amb_load_end}})$ The noise energy of the load calibration with index $I_{\text{amb_load_end}}$.

Process_Calibration_Data continues to adjust the content of the summation variables $\text{Sum}_{\text{E_amb_load}}$ and $\text{Sum}_{\text{N_amb_load}}$ until $\text{Span}_{\text{mid_to_end}}$ is greater than $\text{Span}_{\text{mid_to_begin}}$ or until the $I_{\text{amb_load_end}}$ is the index of the final record in the calibration data buffer. When either of these conditions is met, the algorithm calculates the mean load calibration values using equations (1) and (2). The algorithm applies these mean load calibrations to equation (3) to calculate the representative bandwidth ratio that is associated with time $T_{\text{time_avg}}$.

For each time average calculation, the algorithm checks whether the load calibration with index $I_{\text{amb_load_begin}}$ and the load calibration with index I_{mid} were both acquired when the SeaWinds instrument was operating in Wind Observation Mode. If both elements were acquired when the instrument was in Wind Observation Mode, Process_Calibration_Data sets the flag $\text{Cal}_{\text{mode_consistent}}$ to TRUE.

Process_Calibration_Data uses a slightly different method to adjust the content of the time average sums for the loop back calibrations. The algorithm runs the logic for the loop back calibrations through two iterations. The first iteration applies for the horizontal polarization antenna beam. The second iteration applies for the vertical polarization antenna beam. For each antenna beam, the routine recalculates the time difference between the loop back calibration measure with index I_{mid} and the two loop back calibration measures with indices $I_{\text{loop_back_begin},i_{\text{pol}}}$ and $I_{\text{loop_back_end},i_{\text{pol}}}$.

$$\text{Span}_{\text{mid_to_begin}} = \text{abs}(T_{\text{mid}} - T_{\text{loop_back_begin},i_{\text{pol}}}) \quad (17)$$

$$\text{Span}_{\text{mid_to_end}} = \text{abs}(T_{\text{mid}} - T_{\text{loop_back_end},i_{\text{pol}}}) \quad (18)$$

where

T_{mid} time of the calibration pulse entry that currently represents the midpoint of the time

average buffer
 $T_{\text{loop_back_begin},i_pol}$ time of the loop back calibration pulse with index $I_{\text{loop_back_begin},i_pol}$
 $T_{\text{loop_back_end},i_pol}$ time of the loop back calibration pulse with index $I_{\text{loop_back_end},i_pol}$
 $\text{Span}_{\text{mid_to_begin}}$ time span between the loop back calibration with index $I_{\text{loop_back_begin},i_pol}$ and the midpoint calibration
 $\text{Span}_{\text{mid_to_end}}$ time span between the loop back calibration with index $I_{\text{loop_back_end},i_pol}$ and the midpoint calibration

The algorithm then compares $\text{Span}_{\text{mid_to_begin}}$ to $\text{Span}_{\text{mid_to_end}}$. If $\text{Span}_{\text{mid_to_end}}$ is smaller than or equal to $\text{Span}_{\text{mid_to_begin}}$, the range of loop back calibration measures that apply to the time average must change. The algorithm removes the entry from the summation variables that now falls outside of the data range and then decrements the count of the loop back calibrations in the buffer:

$$\text{Sum}_{E_{\text{loop_back},i_pol}} = \text{Sum}_{E_{\text{loop_back},i_pol}} - \text{Echo}_{\text{loop_back},i_pol}(I_{\text{loop_back_begin},i_pol}) \quad (19)$$

$$\text{Sum}_{N_{\text{loop_back},i_pol}} = \text{Sum}_{N_{\text{loop_back},i_pol}} - \text{Noise}_{\text{loop_back},i_pol}(I_{\text{loop_back_begin},i_pol}) \quad (20)$$

$$\text{Num}_{\text{loop_back},i_pol} = \text{Num}_{\text{loop_back},i_pol} - 1 \quad (21)$$

where

$\text{Echo}_{\text{loop_back},i_pol}(I_{\text{loop_back_begin},i_pol})$ The echo energy over all of the slices of the loop back calibration for the specified beam polarization with index $I_{\text{loop_back_begin},i_pol}$.
 $\text{Noise}_{\text{loop_back},i_pol}(I_{\text{loop_back_begin},i_pol})$ The noise energy of the loop back calibration for the specified beam polarization with index $I_{\text{loop_back_begin},i_pol}$.
 $\text{Num}_{\text{loop_back},i_pol}$ The number of loop back calibrations of the specified polarization incorporated into the summation variables that are used to calculate time averages.

The algorithm then updates the indices of the first and last loop back calibrations that apply to the rolling time average:

$$I_{\text{loop_back_begin},i_pol} = I_{\text{loop_back_begin},i_pol} + 1 \quad (22)$$

$$I_{\text{loop_back_end},i_pol} = I_{\text{loop_back_end},i_pol} + 1 \quad (23)$$

Before `Process_Calibration_Data` adds any new loop back calibrations into the summation variables that are used to calculate the time average, the process checks the relative stability of the input calibration data. The algorithm calculates the net change of the candidate calibration relative to a test value:

$$\text{Net_Change} = (\text{Echo}_{\text{loop_back},i_pol} - \text{Test}_{i_pol}) / \text{Test}_{i_pol} \quad (24)$$

where

$\text{Echo}_{\text{loop_back},i_pol}$ the sum of the echo energy of all twelve slices for the loop back

calibration with the specified polarization associated with index $I_{\text{loop_back_end},i_pol}$.

Test_{i_pol} the test loop back calibration value for the specified beam polarization. With the exception of the first time average iteration, Test_{i_pol} is equal to the mean of the loop back calibrations from the previous iteration.

$\text{Initialize_Cal_Pulse}$ provides the value of Test_{i_pol} for initial iteration.

If the value of Net_Change is less than or equal to $\Delta_{\text{loop_back}}$, the algorithm includes the new loop back calibration element into the sums that are used to calculate the time average, and then increments the count of loop back calibrations:

$$\text{Sum}_{E_loop_back,i_pol} = \text{Sum}_{E_loop_back,i_pol} + \text{Echo}_{\text{loop_back},i_pol}(I_{\text{loop_back_end},i_pol}) \quad (25)$$

$$\text{Sum}_{N_loop_back,i_pol} = \text{Sum}_{N_loop_back,i_pol} + \text{Noise}_{\text{loop_back},i_pol}(I_{\text{loop_back_end},i_pol}) \quad (26)$$

$$\text{Num}_{\text{loop_back},i_pol} = \text{Num}_{\text{loop_back},i_pol} + 1 \quad (27)$$

where:

$\text{Echo}_{\text{loop_back},i_pol}(I_{\text{loop_back_end},i_pol})$ The echo energy over all of the slices of the loop back calibration of the specified polarization with index $I_{\text{loop_back_end},i_pol}$.

$\text{Noise}_{\text{loop_back},i_pol}(I_{\text{loop_back_end},i_pol})$ The noise energy of the loop back calibration of the specified polarization with index $I_{\text{loop_back_end},i_pol}$.

$\text{Num}_{\text{loop_back},i_pol}$ The number of loop back calibrations of the specified polarization incorporated into the summation variables that are used to calculate time averages.

$\text{Process_Calibration_Data}$ continues to adjust the content of the summation variables $\text{Sum}_{E_loop_back,i_pol}$ and $\text{Sum}_{N_loop_back,i_pol}$ until $\text{Span}_{\text{mid_to_end}}$ is greater than $\text{Span}_{\text{mid_to_begin}}$. Once $\text{Span}_{\text{mid_to_end}}$ is greater than $\text{Span}_{\text{mid_to_begin}}$, $\text{Process_Calibration_Data}$ employs equations (3) and (4) to calculate the mean loop back calibration values that are representative of time $T_{\text{time_avg}}$ for the specified polarization. $\text{Process_Calibration_Data}$ then applies these mean values to equation (7) to calculate the representative calibrations for each polarization for time $T_{\text{time_avg}}$.

The algorithm then resets the value of Test_{i_pol} to be equal to the current mean of the echo energy of the loop back calibrations, or $\text{Mean}_{E_loop_back,i_pol}$. The algorithm will use this value of Test_{i_pol} to test the candidate loop back calibration measures for inclusion into the summation variables for the next time average iteration.

For each set of representative calibrations, $\text{Process_Calibration_Data}$ signals a successful calculation of the time averages by setting $\text{Cal}_{\text{usable},i_pol}$ to TRUE. If problems were detected in the calculation of the time averages, $\text{Process_Calibration_Data}$ sets $\text{Cal}_{\text{usable},i_pol}$ to FALSE.

The algorithm begins another iteration by incrementing the midpoint calibration index, I_{mid} . The time average iterations proceed until the index $I_{\text{amb_load_end}}$ represents the last record in the calibration data buffer.

INPUTS

The algorithm reads these input parameters from the Calibration Pulse Product:

| | |
|----------------------------------|---|
| Noise _{amb_load} | Noise energy of an ambient load scatterometer calibration |
| Echo _{amb_load} | Echo energy of an ambient load scatterometer calibration |
| Noise _{loop_back,i_pol} | Noise energy of a loop back scatterometer calibration for the specified beam polarization |
| Echo _{loop_back,i_pol} | Echo energy of a loop back scatterometer calibration for the specified beam polarization |
| T _{cal} | The time associated with the corresponding sequence of calibrations |

Subroutine Initialize_Cal_Pulse determines the initial values of the following input parameters:

| | |
|------------------------------------|---|
| Num _{amb_load} | The number of load calibrations in the time average calculation |
| Num _{loop_back,i_pol} | The number of loop back calibrations in the time average calculation for each polarization |
| I _{mid} | Index of the midpoint calibration |
| I _{amb_load_begin} | Index of the first load calibration in the time average |
| I _{amb_load_end} | Index of the final load calibration in the time average |
| I _{loop_back_begin,i_pol} | Index of the first loop back calibration of the specified polarization in the time average |
| I _{loop_back_end,i_pol} | Index of the final loop back calibration of the specified polarization in the time average |
| Test _{i_pol} | Test value used to eliminate possible outlier values from the distribution of loop back calibrations |
| Sum _{E_amb_load} | The sum of the echo energy over all of the slices for all of the load calibration pulses from index I _{amb_load_begin} to index I _{amb_load_end} . |
| Sum _{N_amb_load} | The sum of the noise energy for all of the load calibration pulses from index I _{amb_load_begin} to index I _{amb_load_end} . |
| Sum _{E_loop_back,i_pol} | The sum of the echo energy over all of the slices for all of the loop back calibration pulses of the specified polarization from index I _{loop_back_begin,i_pol} to index I _{loop_back_end,i_pol} . |
| Sum _{N_loop_back,i_pol} | The sum of the noise energy for all of the loop back calibration pulses of the specified polarization from index I _{loop_back_begin,i_pol} to index I _{loop_back_end,i_pol} . |

OUTPUTS

| | |
|-----------------------------|---|
| T _{time_avg} | the representative time associated with a set of calibration data |
| α | the bandwidth ratio |
| Cal _{i_pol} | a pair of smoothed, representative calibration values, one for each of the beam polarizations |
| Cal _{usable,i_pol} | quality flag that indicates whether the calibration associated with the |

corresponding time for the specified polarization is usable
 $Cal_{mode_consistent}$ flag that indicates whether the instrument mode was consistent over the time span of the calibrations that were applied to the time average

CONSTANTS

All of the following constants are stored in the Level 1B Constants Table:

| | | |
|-------------------------|---|--------|
| Dim_{amb_load} | The prescribed number of elements to calculate a time average of load calibrations. | 800 |
| Dim_{loop_back} | The prescribed number of elements to calculate a time average of loop back calibrations. | 20 |
| Δ_{loop_back} | The maximum change that should appear between a given loop back calibration and the test calibration value. | 1.2 dB |
| $Ratio_{rcv_gain_eu}$ | the SeaWinds instrument receiver gain ratio in engineering units. | 2.917 |

PROCESSING

Step 1: Calculate the mean load calibration and the mean loop back calibration for both beam polarizations that apply to the first pulse in the first telemetry frame of the Level 1A Product using equations (1), (2), (3) and (4).

Step 2: Use equations (6) and (7) to apply the mean calibrations to calculate a bandwidth ratio and representative calibrations for the first pulse in the first telemetry frame of the Level 1A Product.

Step 3: If the time average calculation records no errors or abnormalities, set the value of the Cal_{usable} flag for the appropriate polarization to TRUE.

Step 4: If the calibration with the index $I_{amb_load_begin}$ and the calibration with index I_{mid} were both acquired when the SeaWinds instrument was operating in Wind Observation Mode, set the value of the flag $Cal_{mode_consistent}$ equal to TRUE.

Step 5: Assign the time of the record in the calibration data with index I_{mid} to T_{time_avg} . This variable stores the representative time for each set of time averaged calibrations.

Step 6: This step marks the beginning of the rolling time average calculation. Increment the index of the midpoint calibration in the calibration data buffer as specified in equation (8).

Step 7: Calculate the time span between I_{mid} and $I_{amb_load_begin}$ using equation (9) and the time span between I_{mid} and $I_{amb_load_end}$ using equation (10).

Step 8: If time span $Span_{mid_to_end}$ is smaller than or equal to time span $Span_{mid_to_begin}$, proceeds to step 9. If time span $Span_{mid_to_end}$ is greater than time span $Span_{mid_to_begin}$, proceeds to

step 10.

Step 9: Adjust the contents of the summation variables that are used to calculate the time average of the load calibrations. Removes the calibrations associated with index $I_{\text{amb_load_begin}}$ from the summation variables. Increments the indices $I_{\text{amb_load_begin}}$ and $I_{\text{amb_load_end}}$. Includes the calibrations associated with the new value of index $I_{\text{amb_load_end}}$ into the same summation variables.

Step 10: Calculate a new mean for the echo energy, $\text{Mean}_{E_{\text{amb_load}}}$, and a new mean for the noise energy, $\text{Mean}_{N_{\text{amb_load}}}$, of the load calibrations using equations (1) and (2).

Step 11: Assign the time of the record in the calibration data buffer with index I_{mid} to $T_{\text{time_avg}}$. This variable stores the representative time for each set of time averaged calibrations.

Step 12: Using equation (6), calculate a new bandwidth ratio, α , based on $\text{Mean}_{E_{\text{amb_load}}}$ and $\text{Mean}_{N_{\text{amb_load}}}$.

Step 13: If the calibration with the index $I_{\text{amb_load_begin}}$ and the calibration with index I_{mid} were both acquired when the SeaWinds instrument was operating in Wind Observation Mode, sets the value of the flag $\text{Cal}_{\text{mode_consistent}}$ equal to TRUE.

Step 14: Loop over the two scatterometer beams. The first loop iteration applies to the horizontal polarization beam. The second loop iteration applies to the vertical polarization beam.

Step 15: Calculate the time span between I_{mid} and $I_{\text{loop_back_begin},i_{\text{pol}}}$ using equation (17) and the time span between I_{mid} and $I_{\text{loop_back_end},i_{\text{pol}}}$ using equation (18).

Step 16: If time span $\text{Span}_{\text{mid_to_end}}$ is smaller than or equal to time span $\text{Span}_{\text{mid_to_begin}}$, proceeds to step 17. If time span $\text{Span}_{\text{mid_to_end}}$ is greater than time span $\text{Span}_{\text{mid_to_begin}}$, proceeds to step 19.

Step 17: Adjusts the contents of the summation variables that are used to calculate the time average of the loop back calibrations. Removes the calibrations associated with index $I_{\text{loop_back_begin},i_{\text{pol}}}$ from the summation variables. Decrements the count of the loop back calibrations for the specified polarization, or $\text{Num}_{\text{loop_back},i_{\text{pol}}}$. Increments the indices $I_{\text{loop_back_begin},i_{\text{pol}}}$ and $I_{\text{loop_back_end},i_{\text{pol}}}$.

Step 18: Calculate the net change in value of the loop back calibration with index $I_{\text{loop_back_end},i_{\text{pol}}}$ of the specified polarization against the $\text{Test}_{i_{\text{pol}}}$, the test loop back calibration value using equation (24). With the exception of the first iteration of time average calculations, $\text{Test}_{i_{\text{pol}}}$ is equal to the mean of the loop back calibrations from the previous iteration. $\text{Initialize_Cal_Pulse}$ determines the value of $\text{Test}_{i_{\text{pol}}}$ for the first time average calculation.

Step 19: If the calculated net change is less than or equal to $\Delta_{\text{loop_back}}$, include the loop back

calibration associated with the new value of index $I_{\text{loop_back_end},i_pol}$ into the time average summation variables. Increment $\text{Num}_{\text{loop_back},i_pol}$, the count of loop back calibrations of the specified polarization.

Step 20: Calculate a new mean for the echo energy, $\text{Mean}_{E_loop_back,i_pol}$, and the noise energy, $\text{Mean}_{N_loop_back,i_pol}$, of the loop back calibrations using equations (3) and (4).

Step 21: Reset the value of Test_{i_pol} , the test loop back calibration value, to $\text{Mean}_{E_loop_back,i_pol}$.

Step 21: Apply $\text{Mean}_{E_loop_back,i_pol}$, $\text{Mean}_{N_loop_back,i_pol}$ and a to equation (7) to calculate calibration measures that are representative of time $T_{\text{time_avg}}$.

Step 22: If no errors or abnormalities were detected with the calculation of the time averaged calibrations, set the value of the flag $\text{Cal}_{\text{usable},i_pol}$ to TRUE.

Step 23: If $I_{\text{amb_load_end}}$ is not equal to the index of the last record in the calibration data buffer, return to step 6 for another rolling time average iteration. If $I_{\text{amb_load_end}}$ is equal to the index of the last record in the calibration data buffer, the algorithm terminates.

REFERENCE

- [1] Lou, S. and Liu, Y., SeaWinds/QuikSCAT High and Low Resolution On-orbit Calibration and Noise Subtraction, Interoffice Memorandum 3347-98-019, Jet Propulsion Laboratory, March 27, 1998.

SeaWinds Algorithm Specification

TITLE: Locate Appropriate Calibration Record for Current Pulse
SUBMODULE:
MODULE: Calculate Sigma0 and Kp
CODE: L1B.6.1.3
VERSION: 1.0
DATE: 07/18/01
AUTHOR: Barry Weiss
SUBROUTINE: Get_Cal_Data
LANGUAGE: Fortran 90
HERITAGE: None

L1B.6.1.3 Get_Cal_Data

PURPOSE

This algorithm assigns a calibration record to each scatterometer measurement pulse. The selection of an appropriate calibration record is based on the transmission time of the pulse, and the time associated with each calibration.

BACKGROUND

This algorithm's logic is based upon two critical assumptions. These assumptions are that the pulse data in the Level 1A Product and that the averaged calibration data that are input into this subroutine both appear in temporal order. The Level 1A Processor checks the temporal order of the data, and thus insures that both of these conditions are met.

Upon the first call to Get_Cal_Data, the algorithm initializes the calibration record index, I_{cal} with a value of 1. On all subsequent calls to Get_Cal_Data, the algorithm employs the value of I_{cal} that was established at the completion of the previous call.

The algorithm compares the time of the current pulse against the time of the calibration record with an index of I_{cal} . If the time of the current pulse exceeds the time of the calibration record, the code increments the index I_{cal} , and tests the pulse time against the calibration time identified with the new value of index I_{cal} .

The code continues through this loop until the logic locates a calibration record with a time that follows the time of the current pulse. The function then transfers the contents of that calibration record to the set of output parameters.

$$\alpha_{sel} = \alpha \quad (1)$$

$$Cal_{sel,h_pol} = Cal_{h_pol} \quad (2)$$

$$Cal_{sel,v_pol} = Cal_{v_pol} \quad (3)$$

$$\begin{aligned} \text{Usable}_{i_pol} &= \text{Cal}_{usable,i_pol} & (4) \\ \text{Mode}_{cons} &= \text{Cal}_{mode_consistent} & (5) \end{aligned}$$

The Level 1B Processor does not generate σ_o s for those measurement pulses that Get_Cal_Data associates with bad calibrations or with calibrations that are based on values from inconsistent instrument modes.

INPUTS

The Level 1B Processor calculates the following input parameter in subroutine Compute_Cell_Geometry:

T_{pulse} The time of the transmission of the each measurement pulse

Subroutine Process_Calibration_Data generates a series of calibration records, each of the which contain the following data components:

T_{time_avg} the representative time associated with each set of calibration data
 a the bandwidth ratio
 Cal_{i_pol} a pair of smoothed, representative calibration values for each polarization
 Cal_{usable,i_pol} quality flag that indicates whether the calibration associated with the corresponding time for the specified polarization is usable
 $Cal_{mode_consistent}$ flag that indicates whether the instrument mode was consistent over the time span of the calibrations that were applied to the time average

OUTPUTS

α_{sel} the selected bandwidth ratio
 Cal_{sel,h_pol} the selected calibration value for a horizontal polarization pulse
 Cal_{sel,v_pol} the selected calibration value for a vertical polarization pulse
 $Usable_{i_pol}$ quality flag that indicates whether the selected calibration for the specified polarization is usable
 $Mode_{cons}$ flag that indicates whether the instrument mode was consistent over the span of the time average for the selected calibration

PROCESSING

Step 1: On the first call to this subroutine, initialize the index of the calibration record, I_{cal} to 1. On all subsequent calls to this subroutine, I_{cal} retains the index value identified with the pulse of the previous call.

Step 2: Test the time of the pulse against the time of the calibration record with an index of I_{cal} . If the time of the calibration with index I_{cal} precedes the time of the pulse, proceed to step 3. If the time of the calibration index is the same as or follows the time of the pulse, proceed to step 4.

Step 3: Increment the value of I_{cal} and returns to step 2.

Step 4: Use equations (1), (2), (3), (4) and (5) to assign the values from the calibration record with index I_{cal} to the subroutine output parameters. Exit the subroutine.